DISS. ETH NO. 19420

Modelling and Filtering Almost Periodic Signals by Time-Varying Fourier Series with Application to Near-Infrared Spectroscopy

A dissertation submitted to
ETH Zurich
for the degree of
Doctor of Sciences

by
Ivo Trajković
Dipl. El. Ing. ETH
born on March 3, 1980
citizen of Ennetbaden AG

accepted on the recommendation of
Prof. Dr. Markus Rudin
Prof. Dr. Hans-Andrea Loeliger
PD Dr. Martin Wolf

2010

Abstract

Many processes in nature are oscillatory but not strictly periodic; measuring them results in signals where the period length and the signal shape (slightly) drift over time. Such signals are characterised as "almost periodic".

In this thesis, an intuitive model for almost periodic signals and two algorithms for estimating the model parameters from noisy measurements have been developed and implemented. The model is a Fourier series where the coefficients and the fundamental frequency are allowed to slowly change over time. The model is represented by factor graphs based on which the two algorithms have been derived.

The motivating application is continuous wave near-infrared spectroscopy where near-infrared light of constant intensity penetrates living tissue. The light is scattered and absorbed; a part of it reaches the sensor. The scattering and absorption properties of living tissue, and thus the measured light intensity, vary in particular due to the heartbeat, low frequency oscillations (also Mayer waves), breathing, and neuronal activity. The measured signal is assumed to be the sum of the individual contributions of these physiological processes and noise. Furthermore, the measured signal is often distorted by sporadic movement artefacts.

The aim of the proposed algorithms in near-infrared spectroscopy is to separate the oscillatory component signals (caused by the heartbeat, low frequency oscillations and breathing) from the measured signal, since (i) they disturb the detection of neuronal activity, (ii) they are acquired simultaneously, and their interrelation can be assessed based on the pure version of each of them, and (iii) characterising each of them separately could yield new understandings of the underlying biological processes.

In this thesis, (i) the proposed algorithms and their implementations are described in detail, (ii) it is shown, based on simultaneous near-infrared spectroscopy and electrocardiography measurements, that the proposed algorithms correctly estimate the heartbeat component, (iii) a new application of near-infrared spectroscopy is revealed: estimating the measures of the heart rate variability, (iv) first tests show that the more efficient one of the two algorithms is able to estimate the low frequency oscillations, (v) it is shown by a study that the more efficient one of the algorithms effectively estimates the heartbeat component, (vi) simulation results propose that the neuronal component signal (related to changes in the electrical field potential in the active brain region) should be detectable from the estimates computed with the more efficient one of the algorithms, (vii) a framework has been developed for comparing, evaluating and calibrating algorithms for signal separation and measuring their precision, and (viii) an idea for detecting movement artefacts in the measured signals is outlined.

In conclusion, the proposed algorithms have been successfully applied to near-infrared spectroscopy.

Zusammenfassung

Viele natrliche Prozesse sind oszillierend, jedoch nicht streng periodisch. Das Messen solcher Prozesse ergibt Signale mit zeitlich variierender Periodenlänge und Form. Solche Signale werden als “fast periodisch” eingestuft.

In dieser Arbeit wurden ein intuitives Modell für fast periodische Signale und zwei Algorithmen zur Schätzung der Modellparameter aus verrauschten Messungen entwickelt und implementiert. Das Modell ist eine Fourierreihe, bei welcher die Koeffizienten und die Grundfrequenz zeitlich langsam variieren können. Das Modell wird durch Faktorgraphen dargestellt, mit deren Hilfe die zwei Algorithmen hergeleitet werden.

Die motivierende Anwendung ist Nahinfrarotspektroskopie, wo Licht mit konstanter Intensität lebendes Gewebe durchdringt. Das Licht wird gestreut und absorbiert; ein Teil davon erreicht den Sensor. Die Streuungs- und Absorptionseigenschaften des lebenden Gewebes, und damit die gemessene Lichtintensität, variieren insbesondere wegen dem Herzschlag, der Atmung, der niederfrequenten Schwingungen (auch Mayer-Wellen), und neuronaler Aktivität. Das gemessene Signal wird als die Summe der Beiträge dieser physiologischen Prozesse und des Rauschens angenommen. Ferner ist das gemessene Signal häufig von sporadischen Bewegungsartefakten verzerrt.

Das Ziel der vorgeschlagenen Algorithmen in der Nahinfrarotspektroskopie ist die Trennung der oszillatorischen Komponenten-Signalen (verursacht vom Herzschlag, den niederfrequenten Schwingungen und der Atmung) aus dem gemessenen Signal, da (i) sie die Erfassung neuronaler Aktivitäten stören, (ii) sie gleichzeitig gemessen werden und daher ihre Wechselbeziehungen untersucht werden können aufgrund der puren Version jedes einzelnen von ihnen, und (iii) die Charakterisierung jedes einzelnen von ihnen neue Einsichten in die zugrunde liegenden biologischen Prozesse ergeben könnte.

In der vorliegenden Arbeit (i) sind die vorgeschlagenen Algorithmen und deren Implementierungen ausführlich beschrieben, (ii) wird aufgrund gleichzeitiger Nahinfrarotspektroskopie- und Elektrokardiographie-Messungen gezeigt, dass die vorgeschlagenen Algorithmen die Herzschlagkomponente korrekt schätzen, (iii) wird eine neue Anwendung der Nahinfrarotspektroskopie aufgedeckt: Schätzung der Parameter der Herzfrequenzvariabilität, (iv) zeigen erste Tests die Fähigkeit des Effizienteren der beiden Algorithmen auf, niederfrequente Schwingungen zu schätzen, (v) wird in einer Studie gezeigt, dass der Effizientere der beiden Algorithmen die Herzschlag-Komponente effektiv schätzt, (vi) zeigen Simulationsergebnisse, dass neuronale Komponenten-Signale, welche mit Änderungen im elektrischen Feld in der aktiven Hirnregion zusammenhängen, detektierbar sein sollten aus den geschätzten Signalen des Effizienteren der beiden Algorithmen, (vii) wurde eine Software entwickelt für den Vergleich, die Bewertung und die Kalibirierung von Algorithmen zur Signaltrennung und deren Genauigkeitsmessung und (viii) wird eine Idee zur Erfassung von Bewegungsartefakten im gemessenen Signalen geschildert.

Die vorgeschlagenen Algorithmen wurden erfolgreich in der Nahinfrarotspektroskopie angewandt.

Contents

List of Figures

Chapter 1
Introduction

1.1 Objectives and outline

In continuous wave near-infrared spectroscopy (NIRS), near-infrared light of constant intensity penetrates living tissue. The light is scattered and absorbed; a part of it reaches the sensor. The scattering and absorption properties of living tissue, and thus the measured light intensity, vary in particular due to the heartbeat, low frequency oscillation (LFO), breathing, and neuronal activity. The measured signal is assumed to be the sum of the individual contributions of these physiological processes and noise. The contributions of the heartbeat, the LFO and breathing are called "oscillatory components".

These are the main objectives in this thesis.
Objective 1: An algorithm for separating the oscillatory component signals from measured/raw NIRS signals shall be developed and implemented.
Reasons: The oscillatory component signals (i) pose disturbing effects when detecting neuronal activity, (ii) are acquired simultaneously, and their interrelation can be assessed based on the pure version of each of them, and (iii) characterising each of them separately could yield new understandings of the underlying biological processes.
Covered in Section 3.1 where a model of almost periodic signals and a first algorithm for estimating the model parameters is proposed. In Section 3.2, an implementation of this algorithm is described. In Section 4.1, another parameterisation of the model is proposed which leads to considerably faster computations and lower memory usage compared to the first algorithm. An implementation of this second algorithm is described in Section 4.3

Objective 2: It shall be verified that the algorithms correctly work.
Reasons: self-evident.
Covered in Sections 3.3 and 4.3.4, aspect 4 where the heart's interbeat intervals derived from electrocardiography (ECG) highly correlate with their correspondents from NIRS derived with the two algorithms. Furthermore, the estimated model parameters highly agree between the two algorithms (see Section 4.1.3).

Objective 3: The algorithms shall precisely estimate the heartbeat oscillation.
Reasons: The harmonics of heartbeat oscillations (sharp peaks in Fig. 3.1) are assumed to interfere with the neuronal signal component which is related to changes in electrical field potentials in an active brain region.
Covered in Figures 3.2, 3.8, 4.3 and 4.6 where the sharp peaks are captured with both algorithms. A framework (outlined in Chapter 6) for comparing, calibrating and evaluating algorithms for signal separation has been developed.

This framework may help to quantify the precision of such algorithms.

Objective 4: The algorithm(s) shall be applied in studies.
Reasons: Both algorithms effectively separate the heartbeat oscillation; the second algorithm is appropriate for estimating the so called "residual signals" by estimating and removing the heartbeat oscillation including all slower components from a raw NIRS signal. The residual signals are further analysed, since they contain only noise and potentially the neuronal signal component related to changes in electrical field potentials in an active brain region.
Covered in Section 4.4.

This chapter further (i) outlines relevant aspects of NIRS and signal processing, and (ii) introduces the used terminology. Relevant aspects of factor graphs and message passing algorithms are summarised in Chapter 2. The first algorithm, its implementation and application are described in Chapter 3. The second algorithm, its implementation and application are described in Chapter 4. Chapter 5 outlines an idea for detecting movement artefacts in measured NIRS signals. Chapter 6 outlines the framework for comparing and evaluating algorithms for signal separation.

1.2 NIRS and terminology

NIRS is well reviewed in [1]. This section summarises aspects of NIRS relevant in this thesis. Furthermore, the used terminology is introduced.

There are three modes of NIRS: *frequency domain*, *time domain* and *continuous wave*. In this work, solely the continuous wave instrument MCPII (developed in [2]) was used with a sensor depicted in Fig. 3.9. Each light source (light-emitting diode) sends light with different wavelengths and constant intensity; each detector (photodiode) measures light intensity. Using the modified Beer-Lambert law (introduced in [3]) and at least 2 different wavelengths between 700 nm and 1000 nm enables estimating concentration changes of oxy- and deoxyhemoglobin.

MCPII measures up to 16 source/detector combinations, called "light paths", each with 3 different wavelengths resulting in 48 signals, called "data channels". The term "raw NIRS signal" refers to a single data channel. The sampling rate is 100 Hz per data channel, i.e. every 10 ms, up to 48 samples (one sample from each data channel) are acquired according to a time-multiplexed pattern: the 10 ms are divided into time slots of 416 μs. Either two data channels with the same source are measured simultaneously or one data channel is measured in one time slot (see [2], section 5.2.2). Consequently, data channels with unique sources are measured consecutively within the 10 ms; data channel pairs with a common source are measured simultaneously. Furthermore, it is assumed that within the 10 ms the scattering and absorption properties of the illuminated tissue do not change; thus, all data channels are regarded as "simultaneously measured" even if not true in the strict sense of the time-multiplexing.

The NIRS signal's sample values are proportional to (i) the number of pho-

tons per time unit flowing through the detector / photodiode, (ii) the integral over the spectral sensitivity of the photodiode (section 5.4.5, [4]). The physiology influences these quantities. The heartbeat varies the arterial blood volume; breathing varies the volume of the veins, advances and oxygenates (modifies the chromophore properties of) venous blood; LFOs were first assumed to be caused by rhythmical contractions of small arteries and arterioles, but the underlying mechanism still partly remains unclear [5]; in an active brain region (i) several seconds long, localised concentration changes of oxy- and deoxyhemoglobin modify the blood's chromophore properties, (ii) split second long, even more localised electrical field potentials are correlated with the absorption and scattering properties of the illuminated tissue (see Section 1.3).

A measurement with several light paths contains spatially resolved information from the illuminated tissue; in this context, the terms near-infrared imaging (NIRI) and optical topography (OT) are appropriate. In this work, the term NIRS is used, since each raw NIRS signal is processed separately; spatial resolution is thereby disregarded.

The measured NIRS signal is assumed to be the sum of contribution signals of physiological processes and noise. The contribution signal of a physiological process is called "component". The contribution signals of the heartbeat, the LFO and breathing are called "oscillatory components". A neuronal signal component related to changes in electrical field potentials in an active brain region (see Section 4.4) is called "optical neuronal signal".

In this thesis, two algorithms for separating the oscillatory components in raw NIRS signals have been developed and implemented. The umbrella term for these algorithms is parameter estimation of a model for almost periodic signals (PEMAPS). The implementations are called POETDiscretePhase and POETGaussian with the prefix physiological oscillation estimation tool (POET).

1.3 Optical neuronal signal

In [6], the signals of the membrane potential in a single active neuron and the light scattered on the neurons surface were found to be parallel. In [7], a brain slice was electrically stimulated and illuminated with 3 different wave lengths. The optical signals are concurrent with the signals of local field potentials; this motivates the hypothesis that optical neuronal signals can be measured with NIRS. Although [8–12] report success, [13] reports controversies ([1]).

In experiments investigating whether optical neuronal signals can be measured, the subject is repeatedly stimulated by events which invoke neuronal activity at a specific location in the brain above which the NIRS sensor is placed on the scalp. For example, every second during a 20 s period the subject's eyes are exposed to a flash light; during the next 20 s the subject is not stimulated. This pattern is repeated several times. The magnitude of optical neuronal signals is expected between 0.01% – 0.24% [9] of the intensity of raw NIRS signals. To increase the energy of such a possible, by the flash light triggered, optical neuronal signal compared to the energy of the remaining signal components, the raw NIRS signals are averaged with respect to the flash lights.

Each raw NIRS signal is divided into 1000 ms long segments such that during the stimulation periods the segments are located exactly between the flash lights where an optical neuronal signal is expected. Finally, all signal segments during stimulation are averaged resulting in a mean signal segment. Likewise, all signal segments during the periods with no stimulation are averaged resulting in a second mean signal segment. The two mean signal segments may then be compared or statistically tested. The strong oscillatory components in raw NIRS signals and the several seconds long, stimulus related, concentration changes of oxy- and deoxyhemoglobin usually survive the averaging procedure. PEMAPS is applied before the averaging procedure to estimate and then subtract these components from the raw NIRS signals. The resulting residual signals are assumed to contain only noise and, if at all, the optical neuronal signal.

1.4 Signal separation: overview and qualitative comparison of methods

Many methods for separating mixed signals exist.

- independent component analysis (ICA) [14] assumes that (i) the measured signals $\boldsymbol{x}$ (the entries of $\boldsymbol{x}$ could represent different data channels) arise by an unknown linear mapping $\mathbf{A}$ of unknown source signals $\boldsymbol{s}$ (each entry of $\boldsymbol{s}$ could represent a physiological component), i.e. $\boldsymbol{x} = \mathbf{A} \cdot \boldsymbol{s}$, and (ii) the source signals are statistically independent and, except for one of them at most, have non-Gaussian distributions. Based on these assumptions, the estimate $\hat{\mathbf{A}}$ of $\mathbf{A}$ is calculated such that the estimated source signals $\hat{\boldsymbol{s}} = \hat{\mathbf{A}}^{-1} \cdot \boldsymbol{x}$ are statistically as independent as possible. ICA is not suited for separating the heartbeat component in raw NIRS signals, since (i) the blood propagates differently in upper (arterioles and capillaries) than in lower layers (arteries), and the penetration depth of the measured light depends on the source/detector distance, (ii) the heartbeat component is related to the oxygenated arterial blood, and oxyhemoglobin absorbs light in dependence of its wavelength (figure 3.9, [2]). Hence, each raw NIRS signal features its individual heartbeat component.
  Compared to PEMAPS, this method ignores the properties of almost periodic signals. Moreover, it needs rather long signals to work. The assumption that the source signals have non-Gaussian distributions is problematic, since usually these distributions are a priori unknown.
- empirical mode decomposition (EMD) [15] decomposes the measured signal into a finite number of oscillatory modes, called intrinsic mode function (IMF)s, by their characteristic time scales. The IMFs are derived empirically from the measured signal without using prior knowledge or model (see Section 3.3.5). The sum of a certain subset of IMFs represents an estimate of an oscillatory component. EMD is applied on one signal and thus is suited for separating the components in raw NIRS signals.
  Compared to PEMAPS, this method is simple, fast and can be used with

a high variety of signals; its drawback is that it is fully empirical,and it is questionable how the IMFs should be physically interpreted.

- A traditional band-pass filter (applied with NIRS for example in [16]) passes a certain frequency range in the input signal, all other frequencies are attenuated. To estimate the heartbeat component, for example, the band-pass filter's cutoff frequency f_l is chosen according to the lowest assumed heart rate. The high cutoff frequency f_h is chosen according to the highest assumed heart rate multiplied by the number of harmonics (including the fundamental frequency) K in the heartbeat component. In general, the less noise in the raw NIRS signal, the higher is K. Usually, $3 \leq K \leq 10$, $f_l = 0.8$ Hz, $f_h = 7$ Hz for adults, and $f_l = 1.5$ Hz, $f_h = 14$ Hz for newborns.
  Compared to PEMAPS, this method is simple and fast; its main drawbacks are: (i) to remove the typical sharp peaks in the heartbeat component, f_h must be rather high, implying that high-frequency noise survives the filtering procedure, (ii) the fluctuating nature of physiology (the heart rate doubles within seconds after starting a physical exercise) is modelled by many sinusoids of constant frequency which is unnatural, and (iii) due to window-based processing, different components or the harmonics of one component may spectrally overlap in a window.
- principal component analysis (PCA) [17] assumes that the measured signal arises like in ICA. Whereas ICA aims at maximal statistical independence, PCA aims at maximal uncorrelatedness between the estimated source signals in $\hat{\boldsymbol{s}}$, i.e. the covariance between them shall be 0. Note that, independence implies uncorrelatedness, but not vice versa. PCA is not suited for separating the components in raw NIRS signals for the same reasons as ICA.
  Compared to PEMAPS, this method ignores the properties of almost periodic signals. Moreover, it needs rather long signals to work properly.
- In [18], section III.C, a raw NIRS signal was divided into 10 seconds long segments. The LFO is modelled with a sinusoid of constant amplitude and frequency inside such a segment. The amplitude, frequency and the starting angle of the sinusoid are found by a least-squares fit algorithm.
  Compared to PEMAPS, this method assumes constant frequency and amplitude in a segment which is not realistic, in particular when e.g. the subject's body position changes [19].
- The adaptive filter in [2] (section 6.5.2) or [18], section III.B.finds the average shape of a single beat by matching all heartbeat periods in a raw NIRS signal. The times at which one period ends and the next begins are estimated by detecting the diastolic peaks (as in Fig. 3.12). For each heartbeat period in the raw NIRS signal, the interpolated average shape is stretched or compressed in amplitude and in time to fit the period in the least squares sense.
  Compared to PEMAPS, this method needs signals with many heartbeat periods to find the average shape. Moreover, it ignores the varying signal shape in the heartbeat component.
- singular spectrum analysis (SSA), reviewed in [20], projects a measured sig-

nal onto a set of basis functions called empirical orthogonal functions (EOF)s. The resulting projections are called "principal components". EOFs are derived from the autocorrelation function of the measured signal. An oscillatory component is reconstructed by reverse projecting the sum of a specific subset of principal components. SSA is applied on one signal, thus raw NIRS signals can be filtered with SSA.

Compared to PEMAPS, this method can be used with a high variety of signals; its main drawback is that it assumes the measured signal to be stationary in the weak sense; generally, this does not apply to raw NIRS signals, since their mean and autocorrelation changes over time.

1.5 Raw NIRS signals: assumptions

Like in [2], throughout this thesis, it is assumed that all components in a raw NIRS signal, i.e. heartbeat, LFO, breathing, optical neuronal signals, concentration changes of oxy- and deoxyhemoglobin related to neuronal activity, and noise are additively superposed. The assumption of additivity arises by observing raw NIRS signals. In Fig. 3.11A, for example, the additivity of a LFO and a heartbeat component is illustrated.

The components are assumed to be independent from each other which is not always true. It can easily be verified that the heart rate is coupled to breathing. These dependencies could represent additional prior information, especially when the underlying physiological mechanisms are known. Such information is not incorporated in PEMAPS at this time.

The proposed model for almost periodic signals is motivated exclusively through raw NIRS signals without using any assumptions or knowledge on the underlying physiological mechanisms.

1.6 Raw NIRS signals: movement artefacts

During NIRS experiments, the sensor is sporadically moved due to movements of the subject (especially distinct in newborns) causing large fluctuations in most data channels. After the movement, the sensor couples differently against the skin at a slightly different region and/or layer, and thus the signal-to-noise ratio (SNR) (Signal $\stackrel{\triangle}{=}$ sum of all physiological components, noise $\stackrel{\triangle}{=}$ raw NIRS signal - Signal) and the DC offset in the raw NIRS signal are different than before the movement. Nevertheless, the signals between the movement artefacts are regarded to have origin in one and the same brain region and are analysed as a whole.

If (i) a model of all possible movement fluctuations in raw NIRS signals was given, and (ii) the movement fluctuation could be estimated and subtracted from the raw NIRS signal, then the remaining NIRS signal would not be valid during the time spans of a movement, since a proper skin contact of the sensor is not assured. Conclusively, instead of reconstructing the raw NIRS signal during a movement, the aim is to detect the point in time when the movement occurred as robustly as possible. An idea on how to achieve this is outlined in Chapter 5.

Chapter 2
Factor graphs and message passing

Factor graphs allow a unified approach to a number of topics in coding, signal processing, machine learning, statistics, and statistical physics [21]. This chapter points out aspects of Forney-style factor graphs and message passing algorithms [21] used in PEMAPS.

Factor graphs visualise probabilistic factorisations of the model equation for almost periodic signals (3.2). The following rules hold:

- Edges, connected to two nodes, represent random variables.
- Open half-edges, connected to only one node, represent either (i) known variables, i.e. samples from raw NIRS signals and estimates of variables/model parameters, or (ii) random variables.
- Nodes represent conditional probability density function (PDF)s of the outgoing variables/edges, given the incoming variables/edges. A "hard" link between the variables/edges connected to a node is defined through a function; the PDF of the node is 1 for all argument constellations which fulfil this function; otherwise the PDF is 0.
- Nodes connected to only one edge represent prior PDFs of that edge.

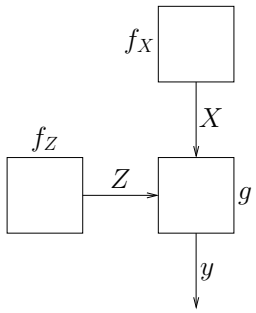

Figure 2.1: Example of a Forney-style factor graph.

In the factor graph in Fig. 2.1, a value y was observed/measured; y is assumed to be the result of two random experiments with unknown outcomes $X = x$ and $Z = z$. The link between y, x and z is assumed to be given by some function $y = g(x, z)$. A frequently-used function is $g(x, z) = x + z$ (represented by the "+"-node). Nodes f_X and f_Z represent prior PDFs of X and Z.

From a more formal point of view, the factor graph in Fig. 2.1 represents the product of all probabilistic factors (nodes), i.e. the joint PDF

$$f_{y,X,Z}(y, x, z) = f_{y|X=x,Z=z}(y, x, z) \cdot f_X(x) \cdot f_Z(z) \tag{2.1}$$

with

$$f_{y|X=x,Z=z}(y, x, z) = \delta(y - g(x, z)) \tag{2.2}$$

with Kronecker delta function $\delta(.)$. Eq. (2.2) expresses $y = g(x, z)$.

One goal in PEMAPS is to estimate an unknown variable assignment by calculating and then maximising its marginal PDF. This is done by summing or

integrating the joint PDF represented by the factor graph over all assignments of values to all unknown variables except for the variable to be estimated. The resulting marginal PDF, decorated with a tilde (e.g. $\tilde{f}_X$), is then maximised over all assignments of the variable to be estimated. For example, to estimate x in Fig. 2.1, the function (2.1) is integrated over z (since z is the only unknown variable assignment beside x). The result is then maximised over all assignments of x. The value of x at the found maximum is the final estimate. This can also be expressed as

$$\hat{x} = \arg\max_x \tilde{f}_X(x) = \arg\max_x \int_z f_{y,X,Z}(y, x, z)dz. \tag{2.3}$$

Evaluating the integral in Eq. (2.3) analytically may be easy in the given example. Usually, the function to be integrated contains many variables, and thus multiple integrals have to be evaluated. Such a "big" marginalisation is shown in Fig. 2.2 (the example is taken from [21]). The factor graph in Fig. 2.2

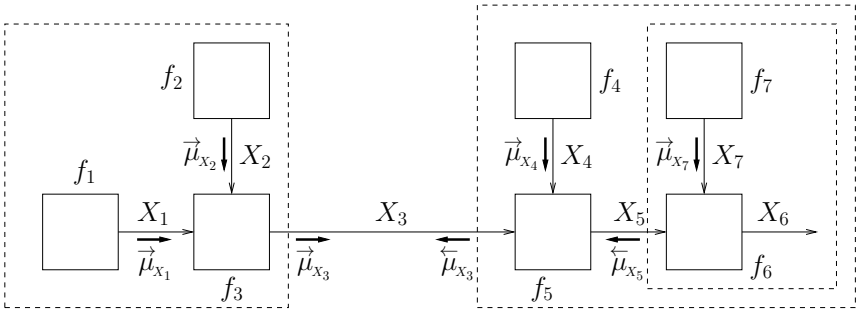

Figure 2.2: Example of a larger factor graph.

represents the product of the PDFs

$$f(x_1, \ldots, x_7) = f_1(x_1)f_2(x_2)f_3(x_3|x_1, x_2)f_4(x_4)f_5(x_5|x_3, x_4)f_6(x_6|x_5, x_7)f_7(x_7) \tag{2.4}$$

The estimate of the assignment x_3, for example, is made as

$$\hat{x}_3 = \arg\max_{x_3} \tilde{f}_3(x_3) \tag{2.5}$$

with

$$\tilde{f}_3(x_3) = \int_{x_1}\int_{x_2}\int_{x_4}\int_{x_5}\int_{x_6}\int_{x_7} f(x_1 \ldots x_7)\,\mathrm{d}x_7\,\mathrm{d}x_6\,\mathrm{d}x_5\,\mathrm{d}x_4\,\mathrm{d}x_2\,\mathrm{d}x_1 \tag{2.6}$$

The function (2.6) can be computed by the sum-product algorithm which splits the "big" marginalisation into a sequence of "small" marginalisations:

$$\tilde{f}_3(x_3) = \vec{\mu}_{X_3}(x_3)\overleftarrow{\mu}_{X_3}(x_3) \tag{2.7}$$

with

$$\vec{\mu}_{X_3}(x_3) = \int_{x_1}\int_{x_2} f_1(x_1)f_2(x_2)f_3(x_3|x_1, x_2)\,\mathrm{d}x_1\,\mathrm{d}x_2 \tag{2.8}$$

and

$$\overleftarrow{\mu}_{X_3}(x_3) = \int_{x_4}\int_{x_5} f_4(x_4) f_5(x_5|x_3, x_4) \overleftarrow{\mu}_{X_5}(x_5)\, \mathrm{d}x_4\, \mathrm{d}x_5 \tag{2.9}$$

with

$$\overleftarrow{\mu}_{X_5}(x_5) = \int_{x_6}\int_{x_7} f_6(x_6|x_5, x_7) f_7(x_7)\, \mathrm{d}x_6\, \mathrm{d}x_7. \tag{2.10}$$

The quantities $\vec{\mu}_{X_3}(x_3)$, $\overleftarrow{\mu}_{X_3}(x_3)$ and $\overleftarrow{\mu}_{X_5}(x_5)$ in Eq. (2.7)–(2.10) may be viewed as summaries of the dashed boxes in Fig. 2.2 which are obtained by integrating over all assignments of each variable inside the box. For discrete-valued variables, the integrals are replaced by sums. Such summaries may be thought of as messages in the factor graph, as shown in Fig. 2.2.

With the trivial summaries/messages $\vec{\mu}_{X_1}(x_1) \triangleq f_1(x_1)$ and $\vec{\mu}_{X_2}(x_2) \triangleq f_2(x_2)$, we can write Eq. (2.8) as

$$\vec{\mu}_{X_3}(x_3) = \int_{x_1}\int_{x_2} f_3(x_3|x_1, x_2) \vec{\mu}_{X_1}(x_1) \vec{\mu}_{X_2}(x_2)\, \mathrm{d}x_1\, \mathrm{d}x_2. \tag{2.11}$$

Similarly, with the trivial summaries/messages $\vec{\mu}_{X_4}(x_4) \triangleq f_4(x_4)$ and $\vec{\mu}_{X_7}(x_7) \triangleq f_7(x_7)$, we can write Eq. (2.9) as

$$\overleftarrow{\mu}_{X_3}(x_3) = \int_{x_4}\int_{x_5} f_5(x_5|x_3, x_4) \vec{\mu}_{X_4}(x_4) \overleftarrow{\mu}_{X_5}(x_5)\, \mathrm{d}x_4\, \mathrm{d}x_5 \tag{2.12}$$

and Eq. (2.10) as

$$\overleftarrow{\mu}_{X_5}(x_5) = \int_{x_6}\int_{x_7} f_6(x_6|x_5, x_7) \vec{\mu}_{X_7}(x_7)\, \mathrm{d}x_6\, \mathrm{d}x_7. \tag{2.13}$$

The messages/summaries (2.11)–(2.13) are formed according to the "sum-product rule": The messages out of some node/factor f_l along some edge X_n is formed as the product of f_l and all incoming messages along all edges except X_n, integrated/summed over all involved variables except X_n.

Another goal in PEMAPS is to estimate all unknown variable assignments by finding the global maximum (assuming that there is one) of their joint PDF. With respect to Fig. 2.2, this can be formulated as

$$(\hat{x}_1, \ldots, \hat{x}_7) = \arg\max_{x_1, \ldots, x_7} f(x_1, \ldots, x_7) \tag{2.14}$$

The estimate of a single variable, e.g. x_3 in Fig. 2.2, is then made as in Eq. (2.5) with

$$\tilde{f}_3(x_3) = \max_{\substack{x_1, \ldots, x_7 \\ \text{except } x_3}} f(x_1 \ldots x_7). \tag{2.15}$$

The function (2.15) can be computed by replacing the integrals in Eq. (2.6)–(2.13) by maximisations. The messages/summaries (2.11)–(2.13) are then formed according to the "max-product rule": The messages out of some node/factor f_l along some edge X_n is formed as the product of f_l and all incoming messages along all edges except X_n, maximised over all involved variables except X_n.

The following additional aspects shall be pointed out.

- The known/measured values of open half-edges such as y in Fig. 2.1 or y_n in Figures 3.4 and 3.5, are simply plugged into the corresponding factors; they are not otherwise involved in the algorithm.
- Open half-edges which represent random variables, such as X_6 in Fig. 2.2, may be thought of being connected to a node with PDF $f(x) = 1$.
- The marginal $\tilde{f}_n(x_n)$ can be obtained for all n by computing two messages, one in each direction, for every edge in the factor graph. The function $\tilde{f}_n(x_n)$ is the product of these two messages as in Eq. (2.7).
- When computing the marginal $\tilde{f}_n(x_n)$ using the sum-product algorithm, all assignment constellations of variables to be integrated/summed over are taken into account regardless of how "probable" they are; the max-product algorithm only considers the constellation for which $\tilde{f}_n(x_n)$ is maximal.
- Messages (i) are scaled conditional probability densities of the underlying edge/variable resulting from the sum-product and the max-product algorithms, (ii) can traverse the edges in both directions.
- When computing estimates by applying the "arg max"-operator on the marginal (as is done everywhere in this thesis), the scale factors of messages are irrelevant; to ensure numerically reliable computations, these scale factors may freely be adapted.

Throughout this thesis, messages are named μ including (i) a superscript arrow indicating the forward ($\vec{\mu}$) or backward ($\overleftarrow{\mu}$) direction with respect to the edge direction and (ii) a subscript (e.g. μ_X) which is the name of the underlying edge (e.g. X).

Two message types are relevant for PEMAPS: discrete messages and Gaussian messages. The former arise by equidistantly sampling a message which is given as a function, e.g. the values of $\mu(\phi) = \lambda e^{-\lambda\phi}$ at $\phi = 0, \Phi, 2\Phi, ...$ with sampling interval Φ represent a discrete message. Each value in the message is called "pixel". When passing discrete messages in a factor graph, all integrals are replaced by sums. To compute a marginal, these sums are evaluated by brute force which is costly. Thus, discrete messages are used when the message functions are complex, and computation rules in a factor graph for the parameters of these functions can hardly be derived.
Gaussian messages $\mu_{\boldsymbol{x}}(\boldsymbol{x}) \propto e^{\frac{1}{2}(\boldsymbol{x}-\boldsymbol{m}_{\boldsymbol{X}})'\mathbf{V}_{\boldsymbol{X}}{}^{-1}(\boldsymbol{x}-\boldsymbol{m}_{\boldsymbol{X}})}$ are fully determined by the mean vector $\boldsymbol{m}_{\boldsymbol{X}}$ and the covariance matrix $\mathbf{V}_{\boldsymbol{X}}$. In addition, the parameters "weight matrix" $\mathbf{W}_{\boldsymbol{X}} \triangleq \mathbf{V}_{\boldsymbol{X}}{}^{-1}$ and "weighted mean" $\mathbf{W}_{\boldsymbol{X}}\boldsymbol{m}_{\boldsymbol{X}}$, including possible superscript direction arrows inherited from the associated message, e.g. $\vec{\mathbf{W}}_{\boldsymbol{X}}$, are used. In [21], computation rules for these parameters in a linear Gaussian

factor graph are given and proved. Such rules allow efficient message computations. Furthermore, in these factor graphs, the sum-product algorithm and the max-product algorithm coincide (up to a scale factor) [21].

Two special nodes in conjunction with Gaussian messages are used in PEMAPS. The node in Fig. 2.3 is equivalent with the "="-node in table 2

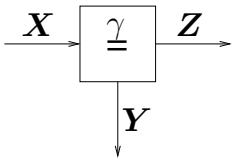

Figure 2.3: The first special node

in [21], except that the significances of the incoming message $\vec{\mu}_{\boldsymbol{X}}$ when calculating $\vec{\mu}_{\boldsymbol{Z}}$, or $\overleftarrow{\mu}_{\boldsymbol{Z}}$ when calculating $\overleftarrow{\mu}_{\boldsymbol{X}}$ respectively, is reduced before the summary proceeds. This is achieved by taking the incoming message to the power of $0 < \gamma < 1$, e.g.

$$\vec{\mu}_{\boldsymbol{X}}(\boldsymbol{x})^{\gamma} \propto \left(e^{\frac{1}{2}(\boldsymbol{x}-\vec{\boldsymbol{m}}_{\boldsymbol{X}})'\vec{\mathbf{V}}_{\boldsymbol{X}}^{-1}(\boldsymbol{x}-\vec{\boldsymbol{m}}_{\boldsymbol{X}})}\right)^{\gamma} = e^{\frac{1}{2}(\boldsymbol{x}-\vec{\boldsymbol{m}}_{\boldsymbol{X}})'\left(\frac{\vec{\mathbf{V}}_{\boldsymbol{X}}}{\gamma}\right)^{-1}(\boldsymbol{x}-\vec{\boldsymbol{m}}_{\boldsymbol{X}})}. \tag{2.16}$$

The only consequence of Eq. (2.16) is that the covariance matrix $\vec{\mathbf{V}}_{\boldsymbol{X}}$ is divided by γ, in other words the weight matrix $\vec{\mathbf{W}}_{\boldsymbol{X}}$ is multiplied by γ. The following computation rules for Gaussian messages in the node in Fig. 2.3 hold (compare with table 2 in [21]).

$$\begin{aligned}
\vec{\mathbf{W}}_{\boldsymbol{Z}} &= \gamma\vec{\mathbf{W}}_{\boldsymbol{X}} + \overleftarrow{\mathbf{W}}_{\boldsymbol{Y}} && (2.17)\\
\overleftarrow{\mathbf{W}}_{\boldsymbol{X}} &= \gamma\overleftarrow{\mathbf{W}}_{\boldsymbol{Z}} + \overleftarrow{\mathbf{W}}_{\boldsymbol{Y}} && (2.18)\\
\vec{\mathbf{W}}_{\boldsymbol{Y}} &= \vec{\mathbf{W}}_{\boldsymbol{X}} + \overleftarrow{\mathbf{W}}_{\boldsymbol{Z}} && (2.19)\\
\vec{\mathbf{W}}_{\boldsymbol{Z}}\vec{\boldsymbol{m}}_{\boldsymbol{Z}} &= \gamma\vec{\mathbf{W}}_{\boldsymbol{X}}\vec{\boldsymbol{m}}_{\boldsymbol{X}} + \overleftarrow{\mathbf{W}}_{\boldsymbol{Y}}\overleftarrow{\boldsymbol{m}}_{\boldsymbol{Y}} && (2.20)\\
\overleftarrow{\mathbf{W}}_{\boldsymbol{X}}\overleftarrow{\boldsymbol{m}}_{\boldsymbol{X}} &= \gamma\overleftarrow{\mathbf{W}}_{\boldsymbol{Z}}\overleftarrow{\boldsymbol{m}}_{\boldsymbol{Z}} + \overleftarrow{\mathbf{W}}_{\boldsymbol{Y}}\overleftarrow{\boldsymbol{m}}_{\boldsymbol{Y}} && (2.21)\\
\vec{\mathbf{W}}_{\boldsymbol{Y}}\vec{\boldsymbol{m}}_{\boldsymbol{Y}} &= \vec{\mathbf{W}}_{\boldsymbol{X}}\vec{\boldsymbol{m}}_{\boldsymbol{X}} + \overleftarrow{\mathbf{W}}_{\boldsymbol{Z}}\overleftarrow{\boldsymbol{m}}_{\boldsymbol{Z}} && (2.22)
\end{aligned}$$

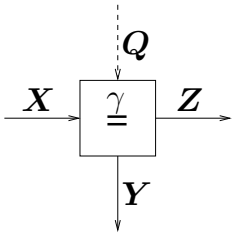

Figure 2.4: The second special node

The node in Fig. 2.4 is equivalent with the node in Fig. 2.3, except that when calculating $\vec{\mu}_{\boldsymbol{Z}}$, the (diagonal) weight matrix of $\vec{\mu}_{\boldsymbol{Q}}$ is added to the weight matrix of $\vec{\mu}_{\boldsymbol{Z}}$, i.e. Eq. (2.17) and Eq. (2.20) become

$$\begin{aligned}
\vec{\mathbf{W}}_{\boldsymbol{Z}} &= \gamma\vec{\mathbf{W}}_{\boldsymbol{X}} + \overleftarrow{\mathbf{W}}_{\boldsymbol{Y}} + \vec{\mathbf{W}}_{\boldsymbol{Q}} && (2.23)\\
\vec{\mathbf{W}}_{\boldsymbol{Z}}\vec{\boldsymbol{m}}_{\boldsymbol{Z}} &= \gamma\vec{\mathbf{W}}_{\boldsymbol{X}}\vec{\boldsymbol{m}}_{\boldsymbol{X}} + \overleftarrow{\mathbf{W}}_{\boldsymbol{Y}}\overleftarrow{\boldsymbol{m}}_{\boldsymbol{Y}} + \vec{\mathbf{W}}_{\boldsymbol{Q}}\vec{\boldsymbol{m}}_{\boldsymbol{Q}}. && (2.24)
\end{aligned}$$

The mean vector $\vec{\boldsymbol{m}}_{\boldsymbol{Q}} \triangleq \mathbf{0}$ and hence $\vec{\mathbf{W}}_{\boldsymbol{Q}}\vec{\boldsymbol{m}}_{\boldsymbol{Q}} = \mathbf{0}$; thus Eq. (2.24) is equivalent with Eq. (2.20).

Chapter 3
An approach to modelling and estimating almost periodic signals

3.1 The approach

Title of the publication
Modelling and Filtering Almost Periodic Signals by Time-Varying Fourier Series with Application to Near-Infrared Spectroscopy

---

Authors and affilations
Ivo Trajkovic [1,3], Christoph Reller [2], Martin Wolf [3], Hans-Andrea Loeliger [2]

[1] Dept. ITET, ETH Zurich, Switzerland, trajkovic@biomed.ee.ethz.ch
[2] Dept. ITET, ETH Zurich, {reller, loeliger}@isi.ee.ethz.ch
[3] Dept. Obstetrics, University Hospital Zurich, {ivo.trajkovic, martin.wolf}@usz.ch

---

Status
Published in the peer-reviewed *Proceedings of the 17th European Signal Processing Conference (EUSIPCO)* pp. 632–636, 2009

Abstract
We propose a new approach to modelling almost periodic signals and to model-based estimation of such signals from noisy observations. The signal model is based on Fourier series where both the coefficients and the fundamental frequency can continuously change over time. This signal model can be represented by a factor graph which we use to derive message passing algorithms to estimate the time-dependent model parameters from the observed samples.

Our motivating application is near-infrared spectroscopy. In this application the observed signal is a superposition of several physiological signals of clinical interest (including, in particular, the arterial pulsation), and we wish to decompose the observed signal into these components. Most of these component signals are almost periodic. We show that the proposed algorithm can be used to extract the arterial pulsation from the measured signal.

3.1.1 Introduction

Many signals in nature are almost periodic. In this paper, we propose a new approach to modelling and to model-based estimation of such signals. Our immediate motivation comes from near-infrared spectroscopy (NIRS), where the observed signals typically look as in Fig. 3.1. The main feature in Fig. 3.1

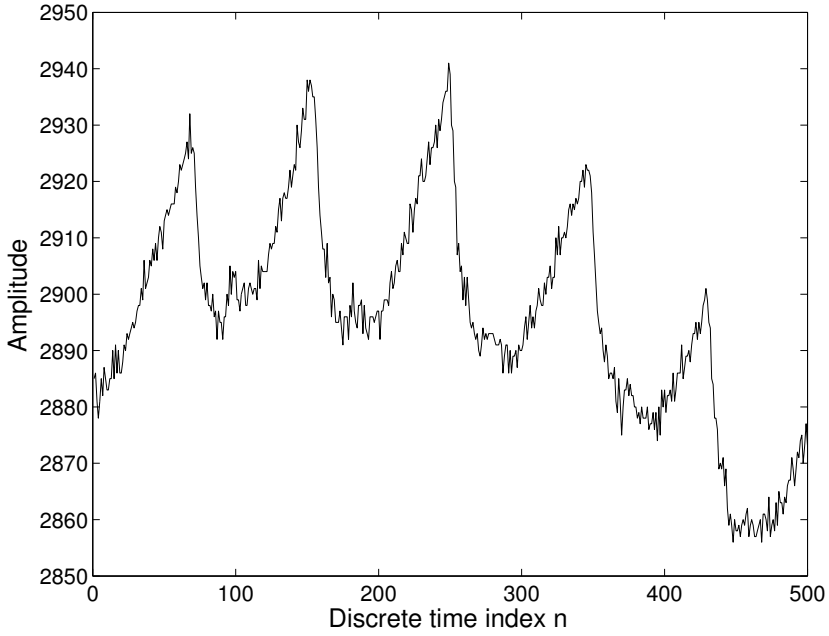

Figure 3.1: A measured NIRS signal with about five periods of the arterial pulsation.

is the arterial pulsation (the heartbeat), which is almost periodic (with a period of about 100 samples in this example). The challenge is to extract the "clean" arterial pulsation and to subtract it from the observed signal in order to make the residual signal available for further analysis (see Section 3.1.2). Note that because of the sharp peaks of the pulses, a simple low-pass filter will not do.

Recall that a periodic signal can be represented by a Fourier series. Specifically, let $x_1, x_2, \ldots$ be the sampled version (with equidistant samples) of a periodic real-valued signal. Then we can write

$$x_n = \mathrm{Re}\left(\sum_{k=0}^{\infty} A_k e^{jkn\Omega}\right) \tag{3.1}$$

with real coefficient A_0, with complex coefficients $A_1, A_2, \ldots$, and with fundamental frequency $\Omega \in \mathbb{R}$. We now propose to model almost periodic signals by changing (3.1) to

$$x_n = \mathrm{Re}\left(\sum_{k=0}^{K} A_{k,n} e^{jk\Theta_n}\right) \tag{3.2}$$

with

$$A_{k,n+1} \approx A_{k,n} \tag{3.3}$$

and

$$\Theta_{n+1} = (\Theta_n + \Omega_n) \bmod 2\pi \tag{3.4}$$

with

$$\Omega_{n+1} \approx \Omega_n. \tag{3.5}$$

Note also that (i) we restrict ourselves to a finite number K of frequencies in (3.2), (ii) we allow the fundamental frequency to continuously change over time, thus Ω is replaced by the time-dependent parameter Ω_n implying that $n\Omega$ in (3.1) is replaced by the parameter Θ_n, from now on called *phase*, and the

relation (3.4) between two temporal consecutive phases is imposed, (iii) we allow the coefficients to continuously change over time, thus the fixed coefficients A_k in (3.1) are replaced by the time-dependent coefficients $A_{k,n}$. The meanings of (3.3) and (3.5) are not formally defined here. These constraints are, however, motivated by the fact that the period length and signal shape, e.g. the heart rate and the beat shape in the arterial pulsation, slightly vary.

Now let $y_1, y_2, \ldots$ be a noisy version of the signal $x_1, x_2, \ldots$. Specifically,

$$y_n = x_n + Z_n \tag{3.6}$$

where $Z_1, Z_2, \ldots$ is discrete-time white Gaussian noise. The main point of this paper is that the parameters $A_{k,n}$ and Θ_n (for $k = 0, 1, \ldots, K$ and for $n = 1, 2, 3, \ldots, N$) can be efficiently estimated from $\boldsymbol{y} = (y_1, \ldots, y_N)$ with a complexity that is linear both in K and in N. An estimate of the "clean" signal $\boldsymbol{x} = (x_1, \ldots, x_N)$ may then be obtained from (3.2).

The proposed approach may be outlined as follows. First, equations (3.2)–(3.6) can be turned into a state space model that can be represented by a factor graph [22, 21]. We then use message passing algorithms in this factor graph to estimate all the parameters. The estimates are optimal in the least-squares sense, i.e. they result in minimal

$$\sum_{n=1}^{N} (y_n - x_n)^2.$$

The soft constraint in (3.3) is handled with adjustable strength by message damping, as will be described in Section 3.1.7 ("$\overset{\gamma}{=}$"-node). The soft constraint in (3.5) is handled with adjustable strength by using prior knowledge about the upper and lower limits of Ω_n, as will be described in Section 3.1.6 ("$\mathcal{I}$"-node).

In contrast to other well-known methods, like Independent Component Analysis (ICA) described in [14] or the traditional band-pass filtering used in [16], our method allows explicit modelling of the almost periodic signal resulting in adaptive filtering. A different adaptive filter, presented in [2], Section 6.5.2, finds an average shape by matching all periods in the arterial pulsation of a long NIRS measurement in one subject. This shape comprises periodicity in the arterial pulsation. The disadvantages of this filter compared to our filter are: (i) it needs a large observation dataset, and (ii) it assumes for every period the same shape, which is unrealistic.

The next section of this paper is about NIRS and shows some experimental results with the proposed algorithm. The algorithm itself is described in Section 3.1.3.

3.1.2 Application to Near-Infrared Spectroscopy

NIRS was described in detail in [2], [1], and [23]. Our experimental data was obtained by the equipment developed and described in [2]. Light from a suitable source is sent through some tissue, where it is scattered and absorbed. Some of the light finds its way to the detector. In living tissue, the intensity of

the detected light varies due to a number of physiological effects reflected in the blood [24]. We here assume that the intensity of the detected light is a linear superposition of several component signals including the arterial pulsation (due to the heartbeat), the respiration, slow oscillations, etc., most of which are almost periodic. We wish to decompose the measured signal (the light intensity) into these component signals, all of which are of clinical interest. In particular, we wish to extract the arterial pulsation and subtract it from the measured signal in order to make the residual signal available for further analysis.

Some results with the proposed new method are shown in Fig. 3.2. Many more experimental results are now available which confirm the validity of the proposed method. This means that our approach will improve the diagnosis capabilities and extend the area of application of NIRS.

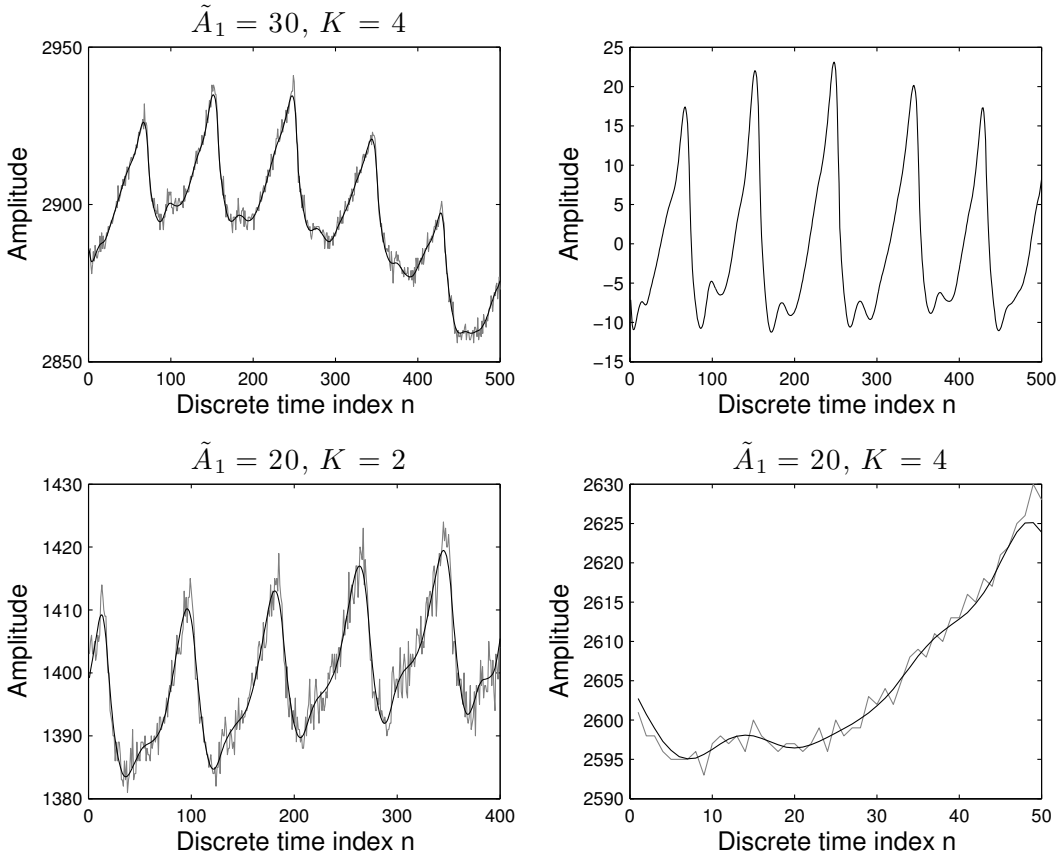

Figure 3.2: Examples of measured NIRS observations (noisy curves) with sampling frequency $f_s = 100$ Hz and corresponding estimated signals (smooth curves). The top right plot shows the estimated arterial pulsation of the signal in the top left plot, which is obtained by subtracting the DC coefficient estimates $(\hat{A}_{0,1}, \ldots, \hat{A}_{0,N})$ from the estimated signal (the smooth curve) in the top left plot. The bottom left plot shows that the algorithm works for observations with rather strong noise. The bottom right plot shows that it also works for short observation datasets (50 samples). In all these examples, the damping parameter γ (see Section 3.1.7) was $\gamma = 0.96$, and convergence was achieved in 3 iterations.

3.1.3 Estimating The Model Parameters

The estimation of the model parameters is structured into several blocks, as will be described in Section 3.1.4. The factor graphs of the two main blocks will be introduced in Section 3.1.5, and the corresponding message passing algorithms will be described in Sections 3.1.6 and 3.1.7.

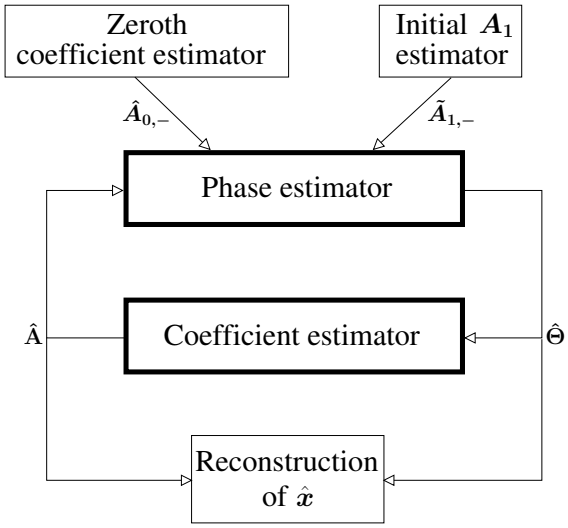

Figure 3.3: The building blocks of the estimation algorithm.

3.1.4 The Building Blocks of the Estimation Algorithm

Given a set of N observed (measured) samples $\boldsymbol{y}$, the objective is to estimate the model parameter vector $\boldsymbol{\Theta} \triangleq (\Theta_1, \ldots, \Theta_N)$ and coefficient matrix

$$\mathbf{A} = \begin{pmatrix} A_{0,1} & \ldots & A_{0,N} \\ \vdots & \ddots & \vdots \\ A_{K,1} & \ldots & A_{K,N} \end{pmatrix}$$

and reconstruct the almost periodic signal by applying the estimates in (3.2). We will use $\boldsymbol{A}_{k,-}$ for the k-th row and $\boldsymbol{A}_{-,n}$ for the n-th column of $\mathbf{A}$.

We propose an iterative, parameter-wise maximum a-posteriori estimation procedure. Based on the observation dataset $\boldsymbol{y}$ and a given estimate $\hat{\mathbf{A}}$ of $\mathbf{A}$ we make estimates $\hat{\Theta}_n$ of Θ_n as

$$\hat{\Theta}_n = \underset{\Theta_n \in [0, 2\pi]}{\arg\max}\ f(\Theta_n \mid \boldsymbol{y}, \hat{\mathbf{A}}), \tag{3.7}$$

where the conditional probability density function f in (3.7) comprises the model (3.2), the relation (3.4) and the constraint (3.5), which is handled with adjustable strength by using prior knowledge about the upper and lower limits of Ω_n, as will be described in Section 3.1.6 ("$\mathcal{I}$"-node).

Likewise we make estimates $\hat{A}_{k,n}$ of $A_{k,n}$ based on the observation dataset $\boldsymbol{y}$, estimates $\hat{A}_{k-1,1}, \ldots, \hat{A}_{k-1,N}, \ldots, \hat{A}_{0,1}, \ldots, \hat{A}_{0,N}$, and $\hat{\boldsymbol{\Theta}}$ from the previous iteration as

$$\hat{A}_{k,n} = \underset{A_{k,n} \in \mathbb{C}}{\arg\max}\ f(A_{k,n} \mid \boldsymbol{y}, \hat{A}_{k-1,1}, \ldots, \hat{A}_{k-1,N}, \ldots, \hat{A}_{0,1}, \ldots, \hat{A}_{0,N}, \hat{\boldsymbol{\Theta}}) \tag{3.8}$$

for increasing k. The function f in (3.8) comprises the model (3.2) and the constraint (3.3), which is handled with adjustable strength by message damping, as will be described in Section 3.1.7 ("$\overset{\gamma}{=}$"-node).

The whole estimation algorithm is split into several building blocks. One block produces the coefficient estimates $\hat{\boldsymbol{A}}_{0,-}$, a second block produces the

initial estimate $\tilde{\boldsymbol{A}}_{1,-}$. The two main blocks produce the coefficient estimate matrix $\hat{\mathbf{A}}$ and the phase estimate vector $\hat{\boldsymbol{\Theta}}$. The last block reconstructs the almost periodic signal $\hat{\boldsymbol{x}}$. The interaction between these blocks is depicted in Fig. 3.3.

The coefficient estimator normally uses the estimates produced by the phase estimator block and vice versa. In the beginning, however, the zeroth coefficient estimator independently estimates the DC component $\hat{\boldsymbol{A}}_{0,-}$ by means of (3.8). Since for this there is no need for $\hat{\boldsymbol{\Theta}}$, this is a one-time procedure based on the observation $\boldsymbol{y}$ only. It can be shown that for this case (3.8) boils down to a first-order low-pass filter. The estimates are then fed to the phase estimator block together with an initial estimate $\tilde{\boldsymbol{A}}_{1,-}$. In almost periodic signals measured with NIRS, most of the signal energy, apart from the DC component $\boldsymbol{A}_{0,-}$ and the noise, lies in the fundamental frequency component $\boldsymbol{A}_{1,-}$. Therefore, when applying our algorithm to NIRS, a first rough estimate of the almost periodic signal is simply a sinusoid. Its magnitude $\tilde{\boldsymbol{A}}_{1,-}$ is calculated by the initial A_1 estimator block in such a way that the sinusoid is of the same energy as the signal $\boldsymbol{y} - \hat{\boldsymbol{A}}_{0,-}$. Based on this estimate the phase estimator is already able to produce a good estimate $\hat{\boldsymbol{\Theta}}$. The latter is handed over to the coefficient estimator, which finally calculates the full set of coefficient estimates $\hat{\mathbf{A}}$.

At this point it is possible to apply entries of $\hat{\boldsymbol{\Theta}}$ and columns of $\hat{\mathbf{A}}$ directly in (3.2) and get a first estimate $\hat{\boldsymbol{x}}$ of the almost periodic signal. The result can, however, be improved by iterating a few times.

3.1.5 Factor Graphs for the Main Building Blocks

The estimations are done by using the sum-product message passing algorithm on factor graphs, which is described in detail in [21].

By using a Fourier series as the model where both the coefficients and the fundamental frequency are time-dependent, the message passing algorithm can be regarded as matching, in the least-squares sense, a sliding discrete-time Fourier series with the observation.

Message passing is applied on the factorial temporal decomposition of the statistical model called factor graph (see [22], Chapter 2). With respect to (3.2)–(3.6) a large factorisation arises which represents the full statistical model under the assumption of additive white Gaussian noise.

There are two factorisations of (3.2) depicted in Figures 3.4 and 3.5. The first is used for estimating $\boldsymbol{\Theta}$ according to (3.7) (phase estimator in Fig. 3.3) and the second is used for estimating $\mathbf{A}$ according to (3.8) (coefficient estimator in Fig. 3.3).

In factor graphs edges represent variables and nodes represent factors. In this paper a factor is either (i) a hard constraint expressing the relationship between two or more variables or (ii) a prior probability density.

Messages are scaled conditional probability densities of the underlying edge, i.e. variable, arising as a result of summary propagation algorithms, in our case the sum-product rule. Messages can traverse the edges generally in both directions and are named μ including an arrow placed above it indicating

the forward ($\vec{\mu}$) or backward ($\overleftarrow{\mu}$) direction with respect to the edge direction.

3.1.6 The Phase Estimator

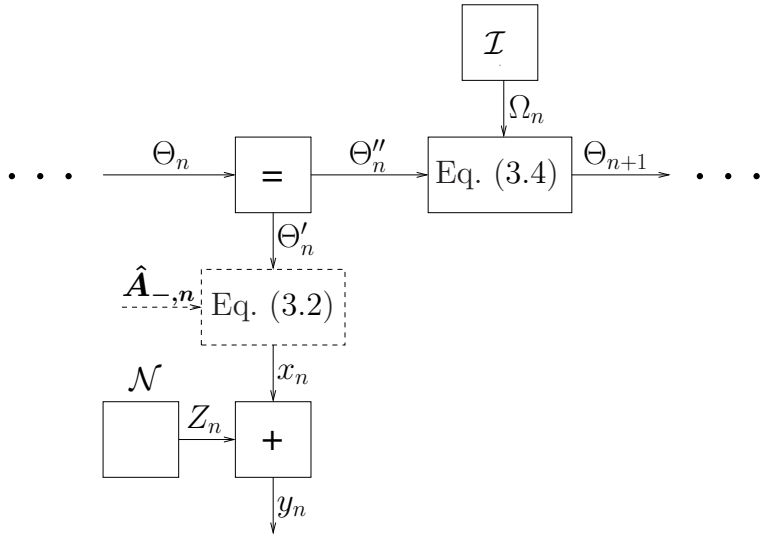

Figure 3.4: Factor graph used for estimation of the phase Θ_n.

The phase estimator makes use of a factor graph containing N consecutive sections one of which is depicted in Fig. 3.4. This means that the outgoing edge Θ_{n+1} of the graph in the figure is at the same time the incoming edge of its right neighbour.

The "Eq. (3.4)"-node represents the mathematical operation in (3.4).

There are two nodes connected to only one output and no input edges. The "$\mathcal{N}$"-node stands for zero-mean white Gaussian noise with variance σ^2, which is added to the clean sample x_n resulting in the observation sample y_n. The "$\mathcal{I}$"-node represents the prior knowledge about the upper and lower limits of the fundamental frequency, i.e. a density $I(\omega)$ with $\omega \in [0, 2\pi]$ which is uniform for $\omega \in [\Omega_{\min}, \Omega_{\max}]$ and 0 elsewhere, dictating that any growth of Θ_n outside the interval $[\Omega_{\min}, \Omega_{\max}]$ is invalid. Depending on the application and the prior knowledge, however, it might be advisable to replace the uniform distribution by a different probability distribution.

Taking the example of the arterial pulsation of humans, the rate of a regular heartbeat takes values between some minimum $H_{\min}$ and some maximum $H_{\max}$. Considering that the acquisition instruments measure with a sampling frequency f_s [Hz] and that during one heartbeat the phase traverses the interval $[0, 2\pi]$, each heart rate H can be assigned to an angle growth Ω according to

$$\Omega(H) = \frac{H \cdot 2\pi}{f_s \cdot 60}.$$

From given $H_{\min}$ and $H_{\max}$ the corresponding angle growth values $\Omega_{\min} = \Omega(H_{\min})$ and $\Omega_{\max} = \Omega(H_{\max})$ can be calculated.

The "="-node expresses cloning of the input variable Θ_n, thus Θ'_n and Θ''_n are clones of Θ_n.

The "Eq.(3.2)"-node represents the mathematical operation in (3.2) with $A_{k,n} = \hat{A}_{k,n}$ for $k = 0, \ldots, K$.

The schedule of the message passing algorithm on the phase estimator factor graph can be stated as follows:

1. For $n = 1, \ldots, N$, calculate $\overleftarrow{\mu}_{\Theta'_n}$ from the observation y_n.
2. For $n = 1, \ldots, N$, first calculate $\vec{\mu}_{\Theta''_n}$ from $\vec{\mu}_{\Theta_n}$ and $\overleftarrow{\mu}_{\Theta'_n}$, then calculate $\vec{\mu}_{\Theta_{n+1}}$ from $\vec{\mu}_{\Omega_n}$ and $\vec{\mu}_{\Theta''_n}$. There is no prior knowledge on Θ_1, thus the message $\vec{\mu}_{\Theta_1}$ is neutral: $\vec{\mu}_{\Theta_1}(\theta) = 1$ for all $\theta \in [0, 2\pi]$.
3. For $n = N, \ldots, 1$, first calculate $\overleftarrow{\mu}_{\Theta''_n}$ from $\vec{\mu}_{\Omega_n}$ and $\overleftarrow{\mu}_{\Theta_{n+1}}$, then calculate $\overleftarrow{\mu}_{\Theta_n}$ from $\overleftarrow{\mu}_{\Theta''_n}$ and $\overleftarrow{\mu}_{\Theta'_n}$. There is no prior knowledge on Θ_{N+1} (or Θ_N respectively), thus $\overleftarrow{\mu}_{\Theta_{N+1}}$ (or equivalently $\overleftarrow{\mu}_{\Theta''_N}$) is neutral: $\overleftarrow{\mu}_{\Theta_{N+1}}(\theta) = \overleftarrow{\mu}_{\Theta''_N}(\theta'') = 1$ for all $\theta, \theta'' \in [0, 2\pi]$.
4. For $n = 1, \ldots, N$, calculate the marginal $\tilde{\mu}_{\Theta''_n} = \vec{\mu}_{\Theta''_n} \cdot \overleftarrow{\mu}_{\Theta''_n}$.
5. For $n = 1, \ldots, N$, calculate the estimate $\hat{\Theta}_n = \arg\max_{\theta_n} \tilde{\mu}_{\Theta''_n}(\theta_n)$.

The messages on the Θ edges can in general not be described by a few parameters. Therefore the messages in the phase estimator are approximated by uniform discretisation.

It might seem more convincing to replace (3.7) with the vector estimate

$$\hat{\boldsymbol{\Theta}} = \arg\max_{\boldsymbol{\Theta} \in [0, 2\pi]^N} f(\boldsymbol{\Theta} \mid \boldsymbol{y}, \hat{\mathbf{A}}).$$

This would lead to changing the sum-product rule to the max-product rule in the phase estimator. In the application with the arterial pulsation in humans measured with NIRS, however, we prefer the sum-product rule because the inherent averaging leads to less overfitting. The latter results from a large heart rate variability ($H_{min} \approx 60$ beats per minute and $H_{max} \approx 180$ beats per minute) implying a wide interval $[\Omega_{min}, \Omega_{max}]$ and thus a very soft constraint (3.5).

3.1.7 The Coefficient Estimator

The coefficient estimator makes use of a factor graph containing N consecutive sections, one of which is depicted in Fig. 3.5. The arrangement of the sections is similar as described in 3.1.6.

The "$\overset{\gamma}{=}$"-node models a central property of almost periodic signals. It is equivalent with the "="-node with the additional feature that the significance of the message coming from the neighbour graph section is somewhat reduced before the summary proceeds. This is done by taking the message to the power of $\gamma \lesssim 1$ which we call "message damping" and which for Gaussian messages results in dividing the variance by γ. This clarifies (3.3) and allows variation of the coefficients from one discrete point in time to the next. During each further summarisation the variance of that message is divided by γ meaning that its significance is exponentially decaying with increasing distance to its original time index.

Furthermore there is the "$c_{k,n}$"-node, which maps some complex coefficient $A_{k,n}$ and phase Θ_n to the k-th harmonic part $\alpha_{k,n}$ of the almost periodic signal

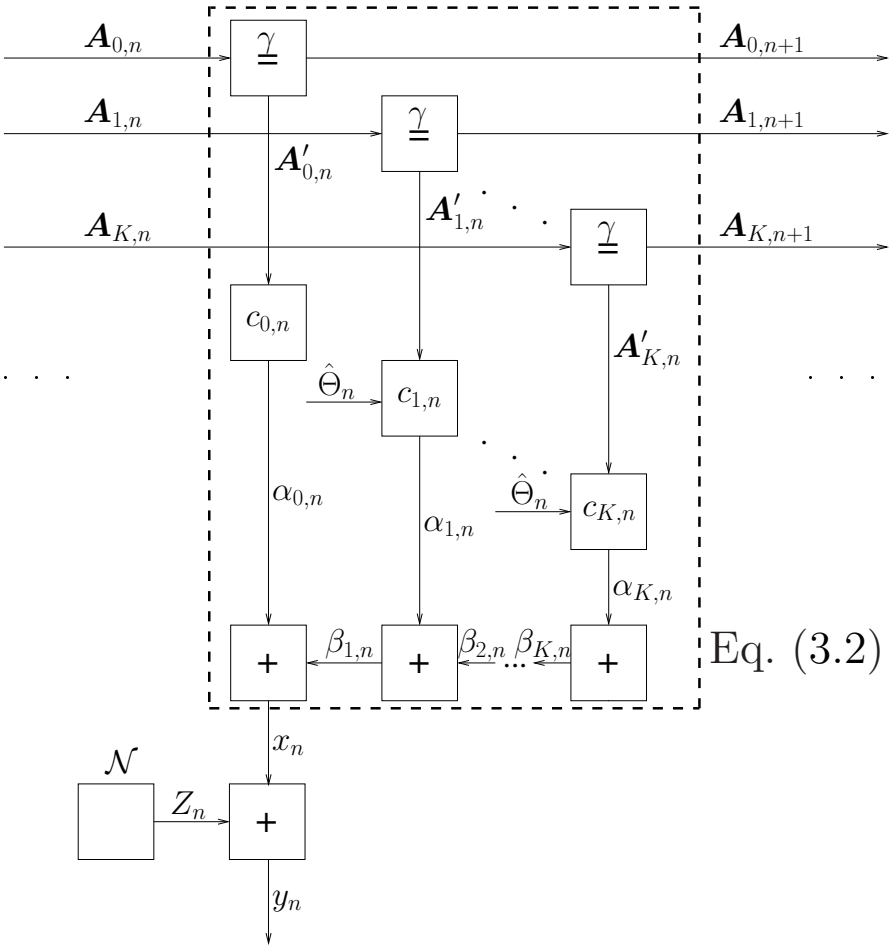

Figure 3.5: Factor graph used for estimation of the coefficient vector $\boldsymbol{A}_{-,n}$.

sample value x_n. Note that the k-th summand of the model equation (3.2) is

$$\begin{aligned}\alpha_{k,n} &\triangleq \mathrm{Re}(A_{k,n} \cdot e^{jk\Theta_n}) \\ &= \mathrm{Re}(A_{k,n}) \cdot \cos(k\Theta_n) - \mathrm{Im}(A_{k,n}) \cdot \sin(k\Theta_n) \\ &= \boldsymbol{A}_{k,n} \cdot \boldsymbol{c}_{k,n} \end{aligned} \tag{3.9}$$

with $\boldsymbol{A}_{k,n} \triangleq (\mathrm{Re}(A_{k,n}), \mathrm{Im}(A_{k,n}))$ and $\boldsymbol{c}_{k,n} \triangleq (\cos(k\Theta_n), -\sin(k\Theta_n))^T$. The summation of all harmonics $\alpha_{k,n}$ for $k = 0, \ldots, K$ results in x_n.

Because Gaussian messages are used and gaussianity is preserved during summary propagation through all nodes in the factor graph in Fig. 3.5 ([21], Chapter V), two parameters, the covariance matrix $\mathbf{V}$ and the mean vector $\boldsymbol{m}$, suffice to describe any message in the graph.

The schedule of the message passing algorithm on the coefficient estimator factor graph can be formulated as follows:

1. Set $k = 0$.
2. For $n = 1, \ldots, N$, calculate sequentially the backward messages $\overleftarrow{\mu}_{x_n}$, $\overleftarrow{\mu}_{\alpha_{k,n}}$ and $\overleftarrow{\mu}_{\boldsymbol{A}'_{k,n}}$ from the observation y_n and estimate $\hat{\Theta}_n$, assuming $\beta_{k+1,n} = 0$ and $\alpha_{m,n} = \hat{\alpha}_{m,n}$ for $m = 0, 1, \ldots, k-1$.
3. For $n = 2, \ldots, N$, calculate $\vec{\mu}_{\boldsymbol{A}_{k,n}}$ from $\vec{\mu}_{\boldsymbol{A}_{k,n-1}}$ and $\overleftarrow{\mu}_{\boldsymbol{A}'_{k,n}}$. There is no prior knowledge on $A_{k,1}$, thus $\vec{\mu}_{\boldsymbol{A}_{k,1}}$ is neutral.
4. For $n = N, \ldots, 1$, calculate $\overleftarrow{\mu}_{\boldsymbol{A}_{k,n}}$ from $\overleftarrow{\mu}_{\boldsymbol{A}_{k,n+1}}$ and $\overleftarrow{\mu}_{\boldsymbol{A}'_{k,n}}$. There is no prior knowledge on $A_{k,N+1}$, thus $\overleftarrow{\mu}_{\boldsymbol{A}_{k,N+1}}$ is neutral.

5. For $n = 1, \ldots, N$, calculate $\vec{\mu}_{\boldsymbol{A}'_{k,n}}$ from $\vec{\mu}_{\boldsymbol{A}_{k,n}}$ and $\overleftarrow{\mu}_{\boldsymbol{A}_{k,n+1}}$.
6. For $n = 1, \ldots, N$, calculate the marginal $\tilde{\mu}_{\boldsymbol{A}'_{k,n}} = \vec{\mu}_{\boldsymbol{A}'_{k,n}} \cdot \overleftarrow{\mu}_{\boldsymbol{A}'_{k,n}}$ and the estimate $\hat{\boldsymbol{A}}_{k,n} = \arg\max_{\boldsymbol{A}_{k,n}} \tilde{\mu}_{\boldsymbol{A}'_{k,n}}(\boldsymbol{A}_{k,n})$.
7. For $n = 1, \ldots, N$, calculate $\hat{\alpha}_{k,n} = \hat{\boldsymbol{A}}_{k,n} \cdot \boldsymbol{c}_{k,n}$ according to (3.9).
8. If $k = K$ all coefficients have been estimated, stop the algorithm or else, increase k and continue on 2.

3.1.8 Results and Conclusion

Many signals in nature are almost periodic. In this paper, we propose a new approach to modelling such signals by means of Fourier series where both the coefficients and the fundamental frequency can continuously change over time. A factor graph representation of such models allows to estimate, with a complexity that is linear both in the number of frequencies K and in the observation length N, the time-dependent model parameters from noisy samples of the signal by means of message passing algorithms.

From a subjective point of view, the resulting estimates, shown in Fig. 3.2, are reasonable already after 3 iterations. We conclude that our approach has been successfully applied to NIRS and thus its usability is shown.

End of publication

3.2 Implementation

In this section, an implementation of the algorithm in Section 3.1 is described. The implementation is called "POETDiscretePhase" and is written in an object-oriented style in C++.

To depict the chronology of processing steps and the interaction between objects in an application, Unified Modelling Language (UML) sequence diagrams are used. The UML notation is introduced in Fig. 3.6.

The diagram of POETDiscretePhase is depicted in Fig. 3.7; the comments on the left establish a connection to Fig. 3.3 and the message passing algorithms in Sections 3.1.6 and 3.1.7.

POETGaussian's "Regularisation"-feature (see Chapter 4) has also been incorporated in POETDiscretePhase for testing purposes. In contrast to POETGaussian, regularising (step 9 in Fig. 3.7) takes place at the end of each "Coefficient estimator"/"Phase estimator" iteration.

The classes in POETDiscretePhase are outlined below. Note the occasional abbreviations: the member method `foo()` in a class `A`, for example, is "`A::foo()`"; a method `bar()` in the global namespace is "`::bar()`". The same holds for objects and variables. Note also that, the term "phase graph" refers to N consecutive sections of the factor graph in Fig. 3.4, and the term "coefficient graph" refers to N consecutive sections of the factor graph in Fig. 3.5.

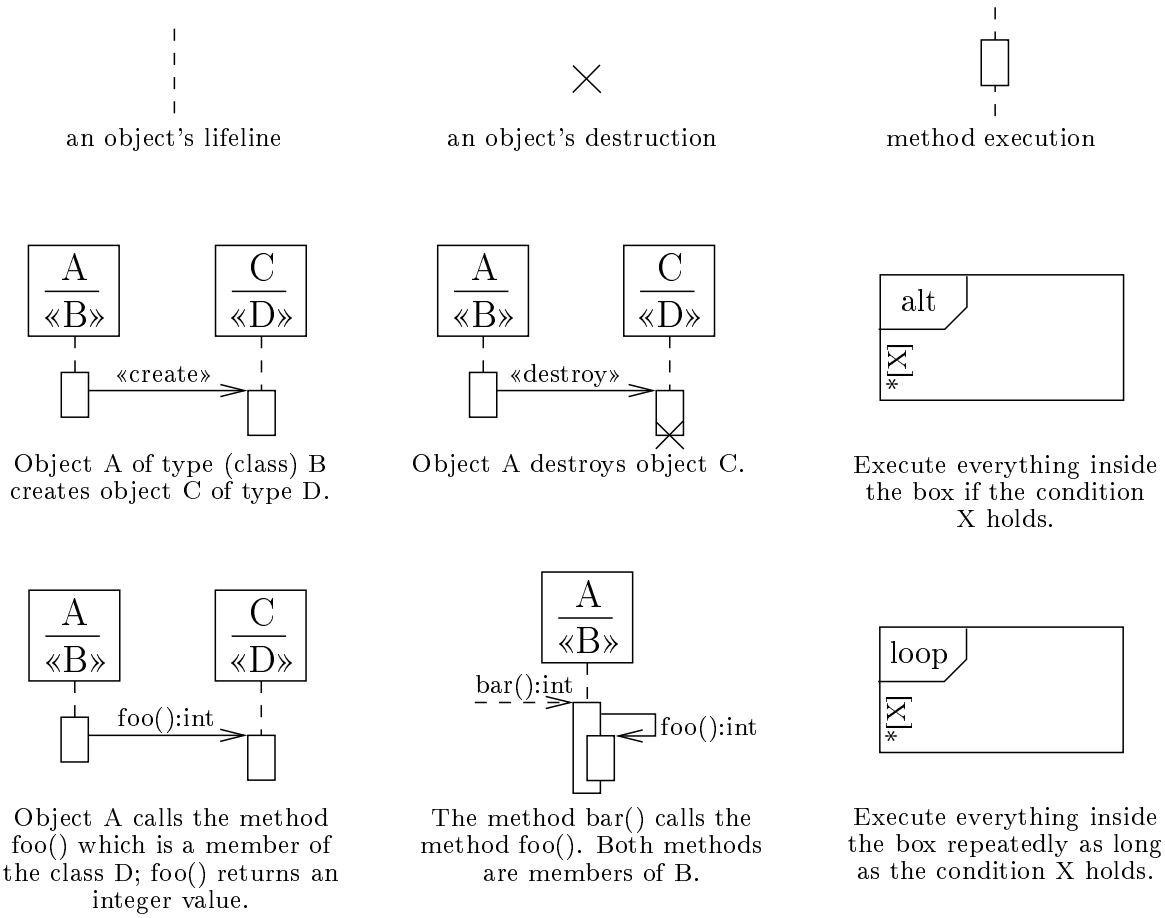

Figure 3.6: Notation in UML sequence diagrams.

3.2.1 The contents of `Core.cpp`

The following objects and variables, defined in the global namespace in `Core.cpp`, should be pointed out.

- `SampleVector sample_buf` holds the raw NIRS signal $\boldsymbol{y}$ (read from file).
- `SampleVector pc_est` holds the reconstructed oscillatory component $\hat{\boldsymbol{x}}$.
- `SampleVector residual` holds the residual signal $\boldsymbol{y} - \hat{\boldsymbol{x}}$.
- `ComplexParameterMatrix ak_est_buf` holds the coefficient matrix $\hat{\mathbf{A}}$.
- `RealParameterVector theta_est_buf` holds the estimates $\hat{\boldsymbol{\Theta}}$.
- `Uint32 discrete_msg_size` holds the number of pixels M in all discrete messages (command line option `-dms`).
- `ParameterDataType var` holds the variance σ^2 of the "$\mathcal{N}$"-node in all factor graphs (command line option `-v`).
- `ParameterDataType omega_min` holds Ω_{min} (command line option `-ol`).
- `ParameterDataType omega_max` holds Ω_{max} (command line option `-or`).
- `std::vector<ParameterDataType> reg_var` holds the regularisation variances ς_k (command line option `-rv`).
- `Uint32 K` holds the number of frequencies K in Eq. (3.2) (command line option `-K`).
- `std::vector<ParameterDataType> msg_coeff_gamma_buf` holds the damping parameters γ (command line option `-g`), one for each $k \in [0, K]$.
- `Uint32 num_of_samples` holds the length N of $\boldsymbol{y}$, i.e. the number of samples read from file (command line option `-N`).

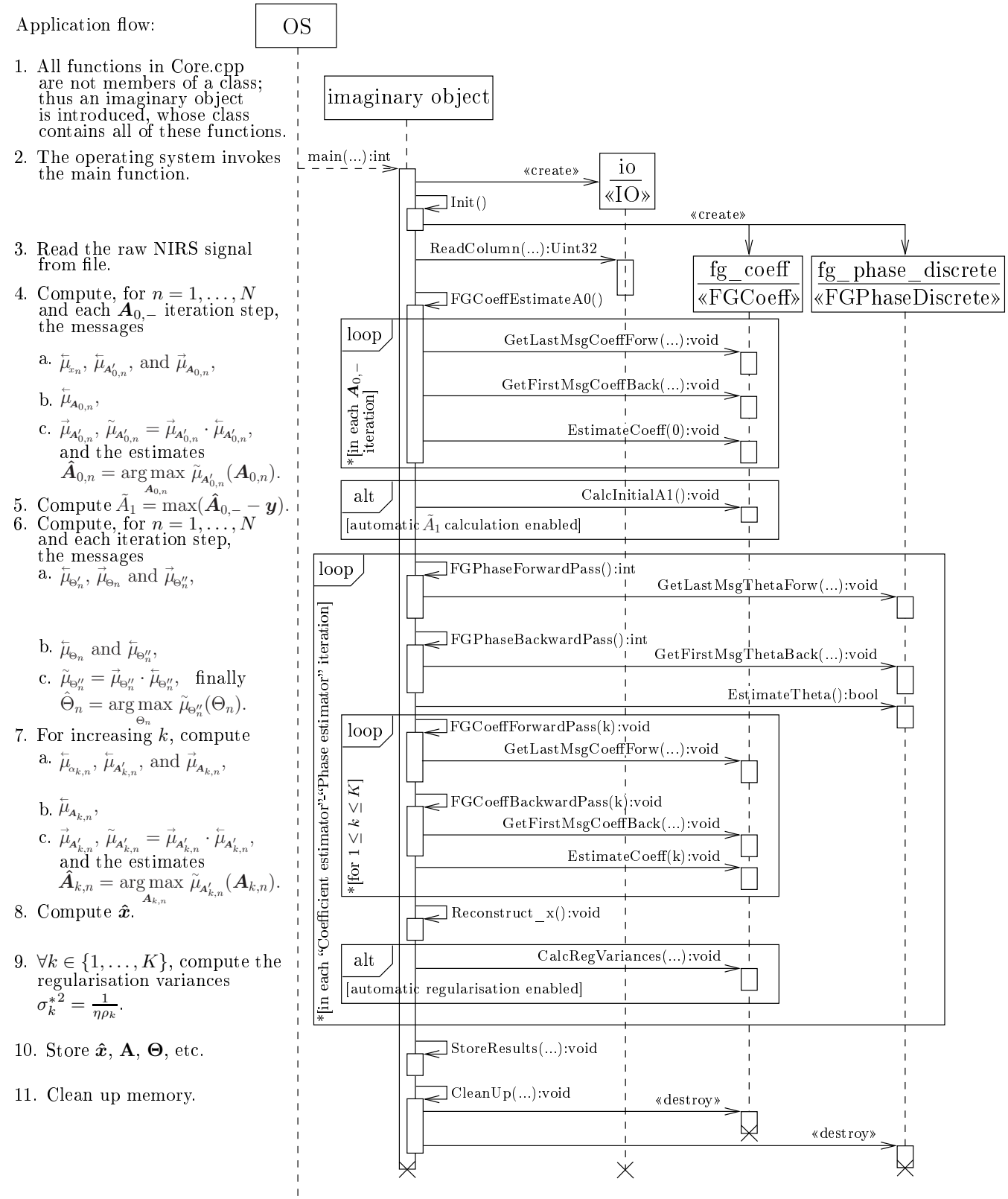

Figure 3.7: The time course of interactions between objects in POETDiscretePhase. Each comment in the "Application flow"-section is positioned roughly next to the corresponding part in the diagram. Rather basic interactions are depicted; several objects and method calls are disregarded.

- `FGPhaseDiscrete* fg_phase_discrete` points to the object representing the phase graph.
- `FGCoeff* fg_coeff` points to the object representing the coefficient graph.

The following methods, defined in the global namespace in `Core.cpp`, should be pointed out.

- `void Reconstruct_x()` reconstructs $\boldsymbol{x}$ (step 8 in Fig. 3.7) from the estimates in `::ak_est_buf` and `::theta_est_buf` and stores it to the buffer `::pc_est`. This method uses the macro `F` described in Section 3.2.6.
- `void CalcResidual()` calculates $\boldsymbol{y} - \hat{\boldsymbol{x}}$ and stores it to the buffer `::residual`.
- `int FGPhaseForwardPass(void* data)` calls `FGPhase::GetLastMsgThetaForw(...)` (described in Section 3.2.4) with arguments (i) `msg_theta_forw_first` referencing a neutral message (see `Discrete::MakeNeutral()` in Section 3.2.4) and (ii) `ret` referencing a temporary object which is discarded.
  The argument `data` is not used but is required, since this method runs as a thread (see Section 3.2.7). An integer is returned for the same reason.
- `int FGPhaseBackwardPass(void* data)` is equivalent with `::FGPhaseForwardPass(...)`, but calls `FGPhase::GetFirstMsgThetaBack(...)`.
- `void FGCoeffEstimateA0()` implements step 4 in Fig. 3.7, to iteratively calculate the coefficient estimates $\hat{\boldsymbol{A}}_{0,-}$ from $\boldsymbol{y}$: in each $\hat{\boldsymbol{A}}_{0,-}$ iteration, the coefficients are reestimated from $\boldsymbol{y} \triangleq \hat{\boldsymbol{A}}_{0,-}$ with $\hat{\boldsymbol{A}}_{0,-}$ from the previous iteration.
  Since estimating $\boldsymbol{A}_{0,-}$ is similar to low pass filtering (see Section 3.1.4), the number of $\hat{\boldsymbol{A}}_{0,-}$ iterations increased by one may be seen as the filter order.
- `void FGCoeffForwardPass(unsigned k)` calls `FGCoeff::GetLastMsgCoeffForw(...)` (described in Section 3.2.2) with arguments (i) `msg_coeff_forw_first` referencing a neutral message (see `Gaussian::MakeNeutral()` in Section 3.2.3), (ii) `ret` referencing a temporary object which is discarded and (iii) the forwarded harmonic index `k`.
- `void FGCoeffBackwardPass(unsigned k)` is equivalent with `::FGCoeffForwardPass(...)`, but calls `FGCoeff::GetFirstMsgCoeffBack(...)`.
- `void StoreResults(bool append, Uint32 start_index, Uint32 end_index)` uses the macros `STORE_VEC`, `STORE_VEC_GIVENFN` and `STORE_MATRIX_STD_COMPLEX` (all described in Section 3.2.5) to store buffers (holding estimates) to files.
  If the argument `append` is "true", the values are appended onto the output files, otherwise the files are overwritten; the arguments `start_index` and `end_index` are marginal access indices which enable to store only selected ranges in buffers.
- `void Init()` (i) instantiates `::pc_est`, `::ak_est_buf`, `::residual` and `::theta_est_buf`, (ii) calculates the variance of the "$\mathcal{N}$"-node in both factor

graphs, if it was not given in the command line (option `-v`), (iii) stores the γ values, one for each $k \in [0, \ldots, K]$, to `::msg_coeff_gamma_buf`, (iv) instantiates the objects `::fg_phase_discrete` (if `::K` > 0) and `::fg_coeff` and (v) sets the regularisation variances in `::fg_coeff` according to `::reg_var`, if automatic regularisation was disabled (command line option `-dt`).

- `int main(int argc, char* argv[])`, alias `main()`, implements the global schedule described in the last three paragraphs in Section 3.1.4. `main()` is overviewed in Fig. 3.7. Furthermore, threads are used to parallelise computing the message sequences $\vec{\mu}_{\Theta_n} \rightarrow \vec{\mu}_{\Theta_n''}$ and $\overleftarrow{\mu}_{\Theta_n} \leftarrow \overleftarrow{\mu}_{\Theta_n''}$ in in Fig. 3.4. This considerably decreases the processing time on processors with multiple cores. Processing times are measured by polling the system's clock (see Section 3.2.7).

The macro `EXECUTE_SIMULTANEOUSLY(fn_ptr1, data1, fn_ptr2, data2)`, defined on the top of `Core.cpp` and used in `main()`, creates and launches immediately two methods as threads running in parallel; `main()` is suspended until they have finished.

3.2.2 The class `FGCoeff`

The class `FGCoeff`, declared in `FGCoeff.hpp` and implemented in `FGCoeff.cpp`, represents the message passing in N consecutive sections of the factor graph in Fig. 3.5; N is thereby the length of the measured raw NIRS signal.

The messages are of type `Gaussian` and are stored in a buffer of type `Gaussian**`; for $k = 0$ they are one-dimensional, since the trend $\boldsymbol{A}_{0,-}$ is real-valued; for $k > 0$ they are two-dimensional.

The following public member methods offer handy services to `main()` (the step numbers refer to the schedule of the message passing algorithm in Section 3.1.7).

- `void GetLastMsgCoeffForw(const Gaussian& msg_coeff_forw_first, Gaussian& ret, unsigned k)` calls `FGCoeff::CalcMsgCoeffForw(...)` (described below) to compute $\vec{\mu}_{\boldsymbol{A}_{k,n}}$ for $n = 1, \ldots, N$. If the estimates $\hat{\boldsymbol{A}}_{k-1,-}$ have not been computed yet, then POETDiscretePhase terminates. If $\overleftarrow{\mu}_{\alpha_{k,n}}$, $c_{k,n}$ and $\overleftarrow{\mu}_{\boldsymbol{A}'_{k,n}}$ have not been computed yet, i.e. `FGCoeff::GetFirstMsgCoeffBack(...)` (described below) was not called previously, the private member methods `CalcMsgAlphaBack(...)`, `Calc_c(...)` and `CalcMsgCoeffUp(...)` (described below) are called first. The argument `msg_coeff_forw_first` references the object holding $\vec{\mu}_{\boldsymbol{A}_{k,1}}$; `ret` references the (instantiated!) object where $\vec{\mu}_{\boldsymbol{A}_{k,N+1}}$ will be stored to. The latter is discarded at the moment, but may be useful in a real time version of POETDiscretePhase.
- `void GetFirstMsgCoeffBack(const Gaussian& msg_coeff_back_last, Gaussian& ret, unsigned k)` calls `FGCoeff::CalcMsgCoeffBack(...)` (described below) to compute $\overleftarrow{\mu}_{\boldsymbol{A}_{k,n}}$ for $n = 1, \ldots, N$. If the estimates $\hat{\boldsymbol{A}}_{k-1,-}$ have not been computed yet, then POETDiscretePhase terminates. If $\overleftarrow{\mu}_{\alpha_{k,n}}$, $c_{k,n}$ and $\overleftarrow{\mu}_{\boldsymbol{A}'_{k,n}}$ have not been computed yet, i.e. `FGCoeff::GetFirstMsgCoeffForw(...)` (described above) was not

called previously, the private member methods `CalcMsgAlphaBack(...)`, `Calc_c(...)` and `CalcMsgCoeffUp(...)` (described below) are called first. The argument `msg_coeff_back_last` references the object holding $\overleftarrow{\mu}_{\boldsymbol{A}_{k,N+1}}$; `ret` references the (instantiated!) object where $\overleftarrow{\mu}_{\boldsymbol{A}_{k,1}}$ will be stored to. The latter is discarded.

- `void EstimateCoeff(unsigned k)` carries out steps 6 and 7 by calling `FGCoeff::CalcMsgCoeffDown(...)` (described below); if $\vec{\mu}_{\boldsymbol{A}_{k,n}}$ and $\overleftarrow{\mu}_{\boldsymbol{A}_{k,n}}$ for $n = 1, \ldots, N$ have not been computed yet, POETDiscretePhase terminates.
- `void CalcInitialA1()` implements step 5 in Fig. 3.7.

In the methods above, $n = 1, \ldots, N$, and k is passed in the method argument `k`. The first three methods terminate POETDiscretePhase, if certain conditions are not fulfilled. This constrains the order in which these methods are called: `EstimateCoeff(...)` must be called after `GetLastMsgCoeffForw(...)` and `GetFirstMsgCoeffBack(...)`, the latter two may be called in an arbitrary order. This mechanism ensures that the schedule of the message passing algorithm is not violated by faulty code in `main()`. Boolean state flags are declared inside `FGCoeff`'s class body for this purpose. By resetting them using `FGCoeff::Reset()` and `FGCoeff::ResetFull()`, the caller may restart the computations in the coefficient graph.

The following private member methods should be pointed out.

- `void CalcMsgAlphaBack(unsigned k)` calculates $\overleftarrow{\mu}_{\alpha_{k,n}}$, according to step 2. If the value of y_n is not-a-number (NaN), $\overleftarrow{\mu}_{x_n}$ is set neutral.
- `void CalcMsgCoeffUp(unsigned k)` calculates $\overleftarrow{\mu}_{\boldsymbol{A}'_{k,n}}$, according to step 2.
- `void CalcMsgCoeffForw(unsigned k)` calculates $\vec{\mu}_{\boldsymbol{A}_{k,n}}$, according to step 3, by calling `Gaussian::RegEqGammaNode(...)`.
- `void CalcMsgCoeffBack(unsigned k)` calculates $\overleftarrow{\mu}_{\boldsymbol{A}_{k,n}}$, according to step 4, by calling `Gaussian::EqGammaNode(...)`, since only the forward pass is regularised (see Section 4.2).
- `void CalcMsgCoeffDown(unsigned k)` calculates $\vec{\mu}_{\boldsymbol{A}'_{k,n}}$, according to step 5, by calling `Gaussian::EqNode(...)`, since the downward passes in the "$\overset{\gamma}{=}$"-node in Fig. 3.5 and in the "="-node are similar (compare table 2 in [21] with Eq. (2.19) and Eq. (2.22)).
- `void Calc_c(unsigned k)` calculates $c_{k,n}$ in Eq. (3.9).

3.2.3 The class `Gaussian`

The class `Gaussian`, declared and implemented in `Gaussian.hpp`, models one- and two-dimensional Gaussian messages and implements the rules for computing such messages in a factor graph. Only the operations/nodes used in the `FGCoeff` class (Section 3.2.2) are implemented.

The member variable `mean` holds $\boldsymbol{m}$; the member variable `variance` holds $\mathbf{V}$. For one-dimensional messages, these parameters are scalars. The weight $\mathbf{W} \triangleq \mathbf{V}^{-1}$ and the weighted mean $\mathbf{W} \cdot \boldsymbol{m}$ are also stored, since the message computations in the "$\overset{\gamma}{=}$"-node in Fig. 3.5 are based on them. Without storing these parameters, they had to be first calculated for all incoming messages in

each "$\overset{\gamma}{=}$"-node from the stored $\boldsymbol{m}$ and $\mathbf{V}$ parameters, and $\mathbf{W} \cdot \boldsymbol{m}$ and $\mathbf{W}$ of the resulting message had to be converted to the corresponding $\boldsymbol{m}$ and $\mathbf{V}$. For the sake of efficiency, the number of such conversions should be kept as low as possible.

The following member methods should be pointed out.

- `void UpdateVariance()` recalculates $\mathbf{V}$ from the current $\mathbf{W}$.
- `void UpdateMean()` recalculates $\boldsymbol{m}$ from the current $\mathbf{W} \cdot \boldsymbol{m}$ and $\mathbf{V}$.
- `void UpdateWeight()` recalculates $\mathbf{W}$ from the current $\mathbf{V}$.
- `void UpdateWeightedMean()` recalculates $\mathbf{W} \cdot \boldsymbol{m}$ from the current $\mathbf{W}$ and $\boldsymbol{m}$.
- `void EqNode(const Gaussian & msg1, const Gaussian & msg2)` implements the "="-node in [21], table 2. The arguments `msg1` and `msg2` reference the input messages. The outgoing message is represented by the `Gaussian` object through which this method is called.
- `void EqGammaNode(const Gaussian & msg1, const Gaussian & msg2, ParameterDataType gamma)` computes Eq. (2.17) and (2.20), or (2.18) and (2.21) respectively. Arguments `msg1` and `msg2` reference the input messages, `gamma` holds γ. The object through which this method is called represents the outgoing message.
- `void RegEqGammaNode(const Gaussian & msg1, const Gaussian & msg2, ParameterDataType gamma, ParameterDataType reg_correction_weight)` computes Eq. (2.23). Arguments `msg1` and `msg2` reference the input messages $\vec{\mu}_X$ and $\overleftarrow{\mu}_Y$, `gamma` holds γ. The object through which this method is called represents $\vec{\mu}_Z$.
- `void MakeNeutral()` makes the message through which this method is called neutral (degenerate Gaussian), i.e. $\mathbf{W} = \mathbf{0}$, $\mathbf{W} \cdot \boldsymbol{m} = \mathbf{0}$, $\boldsymbol{V} = \boldsymbol{\infty}$ and $\boldsymbol{m} = \mathbf{0}$.
- `void ArgMax(BufDataType & ret, unsigned buf_size) const` stores $\arg\max_{\boldsymbol{x}} \mu(\boldsymbol{x})$, which is the mean of the Gaussian message $\mu(\boldsymbol{x})$, to the buffer referenced by `ret`. The argument `buf_size` holds the size of `ret` which is the number of dimensions of $\mu(\boldsymbol{x})$.
- `Gaussian & operator=(const Gaussian & other)` copies all parameters of the message object positioned to the right of the "="-operator (referenced by `other`) to the message object positioned to the left of the "="-operator.

3.2.4 The classes in `FGPhase.hpp`

The class `FGPhase` models message passing in N consecutive sections of the factor graph in Fig. 3.4. Its design was influenced by the hope that other messages than the discrete ones may be used in this graph. Thus, `FGPhase` is independent of the message type; classes modelling mechanisms for passing messages of a specific type shall be derived from it.

Some methods in the class body of `FGPhase` are declared as "virtual" enabling derived classes to override the inherited method by an own one. Virtual methods declared with a terminal "= 0" key, and not implemented inside the class body, must be implemented in the derived class.

The class `FGPhaseDiscrete` models passing of discrete messages in Fig. 3.4. It is derived from `FGPhase` meaning that it inherits all members of `FGPhase`.

The class `Discrete` is the correspondent to the class `Gaussian` for discrete messages.

The step numbers in this section refer to the schedule of the message passing algorithm in Section 3.1.6.

The class `FGPhase`

The message objects in this class must be of generic type. A small class `MsgBuffer` (declared and implemented inside the class body of `FGPhase`) models a buffer for storing such messages. Conclusively, `FGPhase` uses a template for the message type.

The following public member methods should be pointed out.

- `void GetLastMsgThetaForw(const MsgDatatype& msg_theta_forw_first, MsgDatatype& ret)` first calls `FGPhase::PreparePass(...)` (see below) if, for $n = 1, \ldots, N$, $\overleftarrow{\mu}_{\Theta_n}$ and $\overleftarrow{\mu}_{\Theta''_n}$ have not been computed yet. Afterwards, the specific method `CalcMsgThetaForw(...)` (see `FGPhaseDiscrete::CalcMsgThetaForw(...)`) of the class deriving `FGPhase` is called to compute the messages $\vec{\mu}_{\Theta_n}$ and $\vec{\mu}_{\Theta''_n}$.
  The argument `msg_theta_forw_first` references the object holding $\vec{\mu}_{\Theta_1}$ and is forwarded to `CalcMsgThetaForw(...)`; the resulting $\vec{\mu}_{\Theta_N}$ is stored in the message object referenced by the argument `ret`.
- `void GetFirstMsgThetaBack(const MsgDatatype& msg_theta2quotes_back_last, MsgDatatype& ret)` first calls `FGPhase::PreparePass(...)` (see below) if, for $n = 1, \ldots, N$, $\vec{\mu}_{\Theta_n}$ and $\vec{\mu}_{\Theta''_n}$ have not been computed yet. Afterwards, the specific method `CalcMsgThetaBack(...)` (see `FGPhaseDiscrete::CalcMsgThetaBack(...)`) of the class deriving `FGPhase` is called to compute the messages $\overleftarrow{\mu}_{\Theta_n}$ and $\overleftarrow{\mu}_{\Theta''_n}$.
  The argument `msg_theta2quotes_back_last` references the object holding $\overleftarrow{\mu}_{\Theta''_N}$ and is forwarded to `CalcMsgThetaBack(...)`; the resulting $\overleftarrow{\mu}_{\Theta_1}$ is stored in the message object referenced by the argument `ret`.
- `bool EstimateTheta()` returns immediately "false", if neither $\vec{\mu}_{\Theta_n}$ and $\vec{\mu}_{\Theta''_n}$ nor $\overleftarrow{\mu}_{\Theta_n}$ and $\overleftarrow{\mu}_{\Theta''_n}$ for $n = 1, \ldots, N$ have been computed yet. Otherwise, `FGPhase::CalcThetaDistr()` (described below) is called to calculate $\tilde{\mu}_{\Theta''_n}$ for $n = 1, \ldots, N$. Finally, the estimates $\hat{\boldsymbol{\Theta}}$ are computed by calling `Discrete::ArgMax()` in each message object.

`EstimateTheta(...)` must be called after `GetLastMsgThetaForw(...)` and `GetFirstMsgThetaBack(...)`, the latter two may be called in an arbitrary order. This ensures that the schedule of the message passing algorithm is not violated by faulty code in `main()`. Boolean state flags are declared inside `FGPhase`'s class body for this purpose. By resetting them using `FGPhase::Reset()` through the class deriving `FGPhase`, the caller may restart the computations in the phase graph.

The following protected (access only through objects of classes deriving `FGPhase`) member methods should be pointed out.

- `virtual void CalcThetaDistr()` implements the step 4.
- `virtual void PreparePass()` (i) initialises all message buffers when called in the very first "Coefficient estimator"/"Phase estimator" iteration, (ii) calls `CalcMsgThetaUp()`, implemented in the class deriving `FGPhase`, to calculate the messages $\overleftarrow{\mu}_{\Theta'_n}$, and (iii) is invoked in each "Coefficient estimator"/"Phase estimator" iteration by the simultaneously running threads (launched in `main()`) through `FGPhase::GetLastMsgThetaForw(...)` and `FGPhase::GetFirstMsgThetaBack(...)` (described above). A static "mutex" object from the `PI_Mutex` class is used to coordinate these threads. The thread being the first locks the mutex and continues executing. The second thread (i) detects the locked mutex, (ii) waits until the first thread has finished and unlocked the mutex, and (iii) cancels executing. Without (ii), the second thread could start computing the messages $\overleftarrow{\mu}_{\Theta_n}$ and $\overleftarrow{\mu}_{\Theta''_n}$ before the computation of the messages $\overleftarrow{\mu}_{\Theta'_n}$ has finished. This would violate the schedule of the message passing algorithm.

The class `FGPhaseDiscrete`

The following macros should be pointed out.

- `RADIAN_TO_NUM_OF_STEPS(angle_in_radians, discrete_msg_size)` expresses an angle α, given in radians, as pixel index m in a discrete message according to $m = \lfloor \frac{\alpha \cdot M}{2\pi} \rfloor$ with number of message pixels M. The argument `discrete_msg_size` holds M; `angle_in_radians` holds α.
- `NUM_OF_STEPS_TO_RADIAN(angle_in_steps, discrete_msg_size)` does the (inexact but sufficient) reverse transform $\alpha = \frac{m \cdot 2\pi}{M}$.
- `DISCRETE_MSG_NORMALIZE(msg, msg_size)` normalises the discrete message `msg` such that the sum of all pixel values is 1. The argument `msg_size` holds M.

The following member variables should be pointed out.

- `RealParameterMatrix f` holds the values of Eq. (3.2) sampled at $\Theta_n(p) = \frac{2\pi p}{M}$ with $p = 0, \ldots, M-1$, given the estimates $\hat{\boldsymbol{A}}_{-,\boldsymbol{n}}$. `f` is accessed by `f[n][p]` with `n` $= 0, \ldots, N-1$ and `p` $= p$.

The following public member methods should be pointed out.

- `FGPhaseDiscrete(...)` creates an object of type `FGPhaseDiscrete`. First the constructor of the parent class `FGPhase` is called. Then, the member buffer `f` is initialised and the max-product rule is set to be default.
- `void UseSumProductRule()` ensures that the sum-product rule is used.
- `void UseMaxProductRule()` ensures that the max-product rule is used.

The difference between sum-product and max-product only affects the "Eq. (3.4)"-node, since these rules are equivalent for all the other nodes in the phase graph.

The following private member methods should be pointed out.

- `void Fillf()` calculates the values in `FGPhaseDiscrete::f` (see above) by using the macro `F` (see Section 3.2.6).
- `void CalcMsgThetaUp()` implements the step 1. By applying the sum-product rule (see Section 2) in the "Eq. (3.2)"-node in Fig. 3.4, the backward message

$$\overleftarrow{\mu}_{\Theta'_n}(\theta'_n) = \int_{x_n} \overleftarrow{\mu}_{x_n}(x_n)\delta\left(x_n - \mathrm{Re}\sum_{k=0}^{K} \hat{A}_{k,n}e^{jk\theta'_n}\right)\mathrm{d}x_n. \tag{3.10}$$

  The Kronecker delta function represents the hard link induced by Eq. (3.2). Integrating in Eq. (3.10) yields

$$\overleftarrow{\mu}_{\Theta'_n}(\theta'_n) = \overleftarrow{\mu}_{x_n}(\mathrm{Re}\sum_{k=0}^{K} \hat{A}_{k,n}e^{jk\theta'_n}). \tag{3.11}$$

  Applying Eq. (3.6) to the zero-mean Gaussian $\vec{\mu}_{Z_n}(z_n) = \frac{1}{\sqrt{2\pi}\sigma}e^{\frac{z_n^2}{2\sigma^2}}$ yields

$$\overleftarrow{\mu}_{x_n}(x_n) = \frac{1}{\sqrt{2\pi}\sigma}e^{\frac{(x_n-y_n)^2}{2\sigma^2}}. \tag{3.12}$$

  From Eq. (3.11) and (3.12) it follows that

$$\overleftarrow{\mu}_{\Theta'_n}(\theta'_n) = \frac{1}{\sqrt{2\pi}\sigma}e^{\frac{(\mathrm{Re}\sum_{k=0}^{K}\hat{A}_{k,n}e^{jk\theta'_n}-y_n)^2}{2\sigma^2}}. \tag{3.13}$$

  The values of $\overleftarrow{\mu}_{\Theta'_n}$ at $\theta'_n = 0, \Phi, 2\Phi, \ldots, (M-1)\Phi$, with angle interval $\Phi \triangleq \frac{2\pi}{M}$, are computed according to Eq. (3.13) and stored in the buffer `Discrete::body` of the n-th object in the message buffer `FGPhase::msg_theta_up_buf`. If y_n is NaN, $\overleftarrow{\mu}_{\Theta'_n}$ is neutral (see `Discrete::MakeNeutral()`). Note that replacing the integral by a maximisation in Eq. (3.10) would not change Eq. (3.13); thus, in the "Eq. (3.2)"-node, the max-product rule is equivalent with the sum-product rule.
- `void CalcMsgThetaForw(const Discrete& msg_theta_forw_first)` implements the step 2. This method calls `Discrete::EqNote(...)` and `Discrete::PlusNodeSumProd(...)` or `Discrete::PlusNodeMaxProd(...)`. The argument `msg_theta_forw_first` holds $\vec{\mu}_{\Theta_1}$. Some messages are normalised (macro `DISCRETE_MSG_NORMALIZE` described above) to ensure numerically reliable computations.
- `void CalcMsgThetaBack(const Discrete& msg_theta2quotes_back_last)` implements the step 3. This method is similarly implemented to `FGPhaseDiscrete::CalcMsgThetaForw(...)`. The argument `msg_theta2quotes_back_last` references the object which holds $\vec{\mu}_{\Theta''_N}$.

The class `Discrete`

This class models discrete messages. Since a discrete message is fully specified by a scad of pixels, a simple array is used for storing such a message.

In contrast to the `Gaussian` class, not the outgoing message, but the input message to a node is here represented by the object through which a member method is called. There is no special reason for that.

The following member variables should be pointed out.

- `Uint32 omega_min_pixel_idx` holds the pixel index corresponding to $\Omega_{\min}$ in the phase estimator, i.e. $\lfloor \frac{\Omega_{\min} \cdot M}{2\pi} \rfloor$.
- `Uint32 omega_max_pixel_idx` holds $\lfloor \frac{\Omega_{\max} \cdot M}{2\pi} \rfloor$.
- `BodyType body` holds the pixels of the message.

The following member methods should be pointed out.

- `Discrete(unsigned msg_res, ParameterDataType omega_min, ParameterDataType omega_max)` creates a new discrete message object, calculates the member variables `omega_min_pixel_idx` and `omega_max_pixel_idx` by using the `RADIAN_TO_NUM_OF_STEPS` (see Section 3.2.6) macro and initialises the member buffer `body`.
- `void PlusNodeSumProd(Discrete& ret, MsgDirection msg_dir) const` implements the sum-product rule in the "Eq. (3.4)"-node in Fig. 3.4 according to which the forward message

$$\vec{\mu}_{\Theta_{n+1}}(\theta_{n+1}) = \int_0^{2\pi} \int_0^{2\pi} \vec{\mu}_{\Theta_n}(\theta_n) \vec{\mu}_{\Omega_n}(\omega_n) \delta\left(\theta_{n+1} - (\theta_n + \omega_n) \bmod 2\pi\right) \mathrm{d}\theta_n \, \mathrm{d}\omega_n \tag{3.14}$$

is expressed. By applying the rearranged Eq. (3.4) $\Theta_n = (\Theta_{n+1} - \Omega_n) \bmod 2\pi$ and integrating over θ_n in Eq. (3.14), it follows that

$$\vec{\mu}_{\Theta_{n+1}}(\theta_{n+1}) = \int_0^{2\pi} \vec{\mu}_{\Theta_n}\left((\theta_{n+1} - \omega_n) \bmod 2\pi\right) \vec{\mu}_{\Omega_n}(\omega_n) \, \mathrm{d}\omega_n. \tag{3.15}$$

According to the definition of the "$\mathcal{I}$"-node in Section 3.1.6, the message $\vec{\mu}_{\Omega_n}(\omega_n) = C$, where the constant $C \neq 0$ for $\omega_n \in [\Omega_{\min}, \Omega_{\max}]$ and $C = 0$ elsewhere. Using this, Eq. (3.15) reduces to

$$\vec{\mu}_{\Theta_{n+1}}(\theta_{n+1}) = C \int_{\Omega_{\min}}^{\Omega_{\max}} \vec{\mu}_{\Theta_n}\left((\theta_{n+1} - \omega_n) \bmod 2\pi\right) \mathrm{d}\omega_n \tag{3.16}$$

Since all messages in `FGPhaseDiscrete` are discrete, integrals are replaced by sums; thus $\vec{\mu}_{\Theta_{n+1}}(\theta_{n+1})$ in Eq. (3.16) may be approximated as

$$\vec{\mu}_{\Theta_{n+1}}(\theta_{n+1}) \approx \frac{C}{M} \sum_{\omega_n = \Phi \cdot p_{\min}}^{\Phi \cdot p_{\max}} \vec{\mu}_{\Theta_n}\left((\theta_{n+1} - \omega_n) \bmod 2\pi\right) \tag{3.17}$$

with $p_{\min} = \lfloor \frac{\Omega_{\min} \cdot M}{2\pi} \rfloor$, $p_{\max} = \lfloor \frac{\Omega_{\max} \cdot M}{2\pi} \rfloor$ and angle interval $\Phi \triangleq \frac{2\pi}{M}$. Note that $p_{\min}$ is the value in `Discrete::omega_min_pixel_idx`, and $p_{\max}$ is the value in `Discrere::omega_max_pixel_idx`. Adding 2π to the mod2π-argument in Eq. (3.17) allows the substitute $\omega'_n = 2\pi - \omega_n$, and computing the discrete message $\vec{\mu}_{\Theta_{n+1}}$ may finally be formulated as

$$\vec{\mu}_{\Theta_{n+1}}(\theta_{n+1}) \triangleq \sum_{\omega'_n = 2\pi - \Phi \cdot p_{\max}}^{2\pi - \Phi \cdot p_{\min}} \vec{\mu}_{\Theta_n} \left((\theta_{n+1} + \omega'_n) \bmod 2\pi \right). \tag{3.18}$$

Note that in Eq. (3.18), the constant $\frac{C}{M}$ was omitted, since in the end only the argmax of a message is of interest.
Analogously, the discrete message $\overleftarrow{\mu}_{\Theta_n}$ may be derived as

$$\overleftarrow{\mu}_{\Theta_n}(\theta_n) \triangleq \sum_{\omega_n = \Phi \cdot p_{\min}}^{\Phi \cdot p_{\max}} \overleftarrow{\mu}_{\Theta_{n+1}} \left((\theta_n + \omega_n) \bmod 2\pi \right). \tag{3.19}$$

If the argument `msg_dir` (enumerated type) holds the value "FORW", Eq. (3.18) is evaluated and $\vec{\mu}_{\Theta_{n+1}}$ is stored to the object referenced by the argument `ret`; if it holds "BACK", Eq. (3.19) is evaluated and stored. Since these equations only differ in their sum indexing, both are evaluated using the same loop.

- `void PlusNodeMaxProd(Discrete& ret, MsgDirection msg_dir) const` implements the max-product rule in the "Eq. (3.4)"-node in Fig. 3.4. It is equivalent with `Discrete::PlusNodeSumProd(...)`, except that the integrals/sums are replaced by maximisations.
- `void EqNode(const Discrete& other_msg, Discrete& ret) const` implements the sum-product rule in the "="-node in Fig. 3.4 according to which, for example, the message

$$\vec{\mu}_{\Theta''_n}(\theta''_n) = \int_0^{2\pi} \int_0^{2\pi} \vec{\mu}_{\Theta_n}(\theta_n) \overleftarrow{\mu}_{\Theta'_n}(\theta'_n) \delta\left(\theta''_n - \theta_n\right) \delta\left(\theta''_n - \theta'_n\right) \mathrm{d}\theta_n \, \mathrm{d}\theta'_n. \tag{3.20}$$

is expressed. Integrating yields

$$\vec{\mu}_{\Theta''_n}(\theta''_n) = \vec{\mu}_{\Theta_n}(\theta''_n) \overleftarrow{\mu}_{\Theta'_n}(\theta''_n). \tag{3.21}$$

Replacing the integrals in Eq. (3.20) by maximisations as well yields Eq. (3.21) meaning that in the "="-node, the sum-product rule coincides with the max-product rule.
$\vec{\mu}_{\Theta''_n}$ is computed at $\theta''_n = 0, \Phi, 2\Phi, \ldots, (M-1)\Phi$ according to Eq. (3.21) and the pixels are stored in the member `body` of the object referenced by the argument `ret`. If this method is called through the object holding $\vec{\mu}_{\Theta_n}$, the object holding $\overleftarrow{\mu}_{\Theta'_n}$ is referenced by the argument `other_msg`, or vice versa respectively. Messages $\overleftarrow{\mu}_{\Theta_n}(\theta_n) = \overleftarrow{\mu}_{\Theta''_n}(\theta_n) \overleftarrow{\mu}_{\Theta'_n}(\theta_n)$ and $\vec{\mu}_{\Theta'_n}(\theta'_n) = \overleftarrow{\mu}_{\Theta''_n}(\theta'_n) \vec{\mu}_{\Theta_n}(\theta'_n)$ are computed analogously.

- `void MakeNeutral()` sets the value of each message pixel to $\frac{1}{M}$.
- `Discrete & operator=(const Discrete & other)` copies all parameters of the message object positioned to the right of the "="-operator (referenced by `other`) to the message object positioned to the left of the "="-operator. If the target object's pixel buffer is of different size than the source object's pixel buffer, the former is first resized before being filled with new values.
- `ParameterDataType ArgMax() const` finds the pixel with the maximum value by using the macro `FINDMAX_IN_VEC` (see Section 3.2.6) and returns its index.

Discrete messages are used, since passing the input message of type

$$\vec{\mu}_{\Theta_n}(\theta_n) \propto e^{\frac{(\mathrm{Re}\sum_{k=0}^{K}\hat{A}_{k,n}e^{jk\theta_n}-y_n)^2}{2\sigma^2}}$$

(according to Eq. (3.13)) through the "Eq. (3.4)"-node in Fig. 3.4 yields an output message of different type. This can be verified by evaluating the sum in Eq. (3.18) numerically, since it can hardly be done analytically.

3.2.5 The class `IO`

The class `IO` defines read/write access from/to comma-separated values (CSV) files. Raw NIRS signals are read from and signal estimates are written to such files.

Function templates are used to keep the method calls independent of data types, i.e. only one method is written for arbitrary argument types. This improves code readability and keeps the code short. At compile time, the C++ compiler locates the calls to a function template and determines the argument types; hence the corresponding method declaration is generated at compile time, and no independent object file can be generated from the `IO` class. Thus, the latter is declared and implemented in the header file `IO.hpp`.

The following member methods of `IO` should be pointed out.

- `bool WriteValue(const ValueDataType& value, const string& separator)` appends the value of the argument `value` followed by the separator string in the argument `separator` to the CSV file.
  This method is called by `::StoreResults(...)` in `Core.cpp` (through the macro `STORE_MATRIX_STD_COMPLEX` defined in `IO.hpp`) to write complex-valued (`std::complex`) coefficient estimates to a CSV file.
- `bool WriteValues(const VectorDataType& buf, Uint32 len, Uint32 start_index, Uint32 end_index)` stores the vector referenced by `buf` as a column to the CSV file. The argument `len` holds the number of entries in the vector, `start_index` is the index of the first entry in the vector to be written, and `end_index` is the index of the last entry in the vector to be written.
  This method is called by `::StoreResults(...)` in `Core.cpp` (through the macros `STORE_VEC` and `STORE_VEC_GIVENFN` defined in `IO.hpp`) to store buffers to a CSV file.

- `Uint32 ReadColumn(vector<ValueDataType>& buf, Uint32 first_row_index, Uint32 num_of_rows_to_read, Uint32 col_index)` reads the `col_index`-th column and `num_of_rows_to_read` rows, beginning from the `first_row_index`-th, from a CSV file into the (empty) Standard Template Library (STL) vector referenced by `buf`. If `num_of_rows_to_read` is 0, all rows are read.
  This method (i) returns the number of read rows, (ii) ignores the first row if it contains letters, (iii) is able to recognise NaN strings and to store the corresponding value in the buffer, and (iv) is called by `main()` to read raw NIRS signals.

The `bool`-returning methods above return "false", if the file could not be opened; otherwise they return "true".

The macros defined at the beginning of `IO.hpp` integrate the instructions which are necessary (from the point of view of the classes which demand the input/output (IO) services) for an IO access. Each macro contains the call to the corresponding member method of `IO`. The following macros should be pointed out.

- `STORE_VEC(buf, len, start_index, end_index, append)` instantiates an `IO` object, opens the file in append mode if the argument `append` is true, otherwise the file is overwritten or created. The file's name corresponds to the buffer name passed as the argument `buf`. Finally, `IO::WriteValues(buf, len, start_index, end_index)` is called; the success of the IO operation is reported through `std::cout`. If the IO operation was not successful, POETDiscretePhase terminates.
- `STORE_VEC_GIVENFN(buf, m, filename, start_index, end_index, append)` is similar to `STORE_VEC`, except for explicit file naming through the argument `filename`.
- `STORE_MATRIX_STD_COMPLEX(buf, num_of_columns, start_row_index, end_row_index, append)` instantiates an `IO` object and opens a file like e.g. `STORE_VEC`. The magnitude and the angle of each complex value in the buffer `buf` with row index `n` $\in$ [`start_row_index`, ..., `end_row_index`] and column index `k` $\in [0, \ldots,$ `num_of_columns-1`] are written to the file. They are separated by a semicolon. `IO::WriteValue(buf[n][k], string(";"))` is called, or `IO::WriteValue(buf[n][k], string("\n")` respectively, after `num_of_columns` values in a row have been written.

3.2.6 Data types and macros defined in `defines.hpp`

Data types, macros and constants are defined in `defines.hpp`. Primitive data types are preferred, since they grant lower memory usage and faster access when accessing large buffers holding discrete messages.

The following macros should be pointed out.

- `FINDMAX_IN_VEC(vec, N, max, max_idx)` finds the maximum entry in the vector `vec` of length `N` and assigns its index to `max_idx` and its value to `max`.

- `F(ak, theta, x, K)` evaluates Eq. (3.2) with arguments `ak`=$A_{k,n}$, `theta`=Θ_n, `x`=x_n and `K`=K.

3.2.7 Platform independent threads and system clock retrieval

The PI library, written and described in [25] (section D.2), provides a small set of specific methods for running threads, measuring time, accessing network sockets and manipulating the file system. The library runs on Linux, Mac OSX and Windows; it is slim implying that no large overhead is brought in compared to, for example, Simple DirectMedia Layer (SDL).

POETDiscretePhase uses methods for running threads and measuring time from the PI library, in particular the following ones.

- `PI_Thread::PI_Thread(int (*fn)(void*), void* data)` creates a thread of type `PI_Thread`. The argument `fn` is the pointer to a method of type `int mythread(void* data)` to be executed as a thread. The latter is launched immediately. The argument `data` is passed to the thread method.
- `void PI_Thread::Wait(int* status)` waits for a thread to finish. The return code of the thread method is placed in the field pointed by `status`, if `status` is not NULL.
- `unsigned int PI_GetMilliseconds()` returns the number of milliseconds since this method was first called.
- `PI_Mutex::PI_Mutex()` creates a mutex object. Mutexes are used to coordinate threads as described in Section 3.2.4, method `FGPhase::PreparePass()`.
- `void PI_Mutex::Lock()` tries to lock the mutex. If it is already locked, this method waits until another thread unlocks the mutex. Finally this method locks the mutex itself.
- `void PI_Mutex::TryLock()` does the same but returns immediately if the mutex is already locked.
- `void PI_Mutex::Unlock()` unlocks the mutex.

The classes `PI_Thread` and `PI_Mutex` are declared in `pi_threads.hpp` and implemented in `pi_threads.cpp`. The method `::PI_GetMilliseconds()` is declared in `pi_time.hpp` and implemented in `pi_time.cpp`.

3.2.8 The methods in `helper.cpp`

The following handy methods are used in `Core.cpp`, implemented in `helper.cpp` and declared in `helper.hpp`.

- `ParameterDataType GetMean(const SampleVector& sig, Uint32 N)` returns the mean of all entries in the vector of length `N` referenced by `sig` with an iterative procedure which prevents data type overflows.
- `ParameterDataType GetVar(const SampleVector& sig, Uint32 N)` returns the sample variance $\frac{1}{N-1}\sum_{n=1}^{N}(x_n - \bar{x})^2$ with an iterative procedure which prevents data type overflows. The vector referenced by `sig` holds the values x_n, has mean $\bar{x}$ and is of length `N`.

3.2.9 Compiling and running POETDiscretePhase

POETDiscretePhase is compiled by changing to its root directory and executing `make` in the bash console under Linux/OSX or `Windows\make.exe` in the command prompt under Windows. `make opt` produces considerably faster binaries by using the compiler options `-march=native`, `-mtune=native` and `-O3`. The former two options require a recent GNU C++ compiler (version > 4.1).

The GNU C++ compiler and the libraries for compiling POETDiscretePhase under Windows are from the free "Dev-C++"-project[1]. All files are located in `./Windows/` (relative path to POETDiscretePhase's root directory).

The following limitations and practical aspects should be pointed out.

1. `./poet_discrete_phase -h` lists descriptions of all command line options. The relevant ones in this section are: `-K` sets K, `-cg0` sets γ for $k = 0$, `-cgh` sets γ for $k > 0$, `-or` sets $\Omega_{\max}$, `-v` sets σ^2 in all factor graphs.
2. LFOs cannot be estimated with POETDiscretePhase, since a typical message resolution of $M = 500$ has an angle step $\Phi = \frac{2\pi}{500} = 0.013$ rad. which is higher than LFO-typical phase increases between $\Omega_{\min} = 0.0047$ rad. and $\Omega_{\max} = 0.0075$ rad.. Increasing the message resolution would drastically increase processing time and memory usage which are high already with $M = 500$ (see Section 3.3.8).
3. Compared to Linux, the Windows binary's processing time is approximately doubled; the reasons for that were not investigated yet.
4. The following command line, mostly working well with the heartbeat component in adults, was intuitively and empirically determined: `./poet_discrete_phase -ifn /path/to/foo.csv -v 20000 -K 7 -cg0 0.97 -cgh 0.945 -ci 18 -dr -or 0.125`, where the option `-ci 18` is replaced by the real column index (starting from 0), `path/to/foo.csv` is replaced by the real measurement file, and `-K 7` is chosen higher for low-noise NIRS signals and lower for noisy signals. To use regularisation, replace `-K 7` by `-K 15`, remove `-dr`, and add `-rs 0.0005`.
5. Increasing `-K` considerably increases memory usage and processing time. If `-K` is too high and no regularisation is used, the algorithm gradually fits noise.
6. Disabling threads on single-core machines (option `-dt`) reduces context switch related overhead and allows faster computation.
7. The default rule is max-product; option `-us` switches to sum-product.
8. The Matlab® script `./Matlab/load_stored_buf.m` loads the resulting files, i.e. `./pc_est.txt` containing $\hat{\boldsymbol{x}}$, `ak_est_buf.txt` containing $\hat{\mathbf{A}}$, `theta_est_buf.txt` containing $\hat{\boldsymbol{\Theta}}$, and `observation.txt` containing $\boldsymbol{y}$ (a copy of the processed raw NIRS signal), into the Matlab® workspace.
9. The file `./README` contains well working command lines tested with measurement files in `./RawNirsSignals/`.
10. The variance σ^2 of the "$\mathcal{N}(0, \sigma^2)$"-nodes in all factor graphs refers to the variance of the noise in the raw NIRS signal. It does not greatly influence

[1] http://www.bloodshed.net/devcpp.html

the results, however, with lower values (e.g. $\sigma^2 < 100$) the Phase estimator will rather overfit producing thereby less smooth estimates. Note that, σ^2 influences the numerical stability particularly in the phase graph. If `./pc_est.txt` contains NaN's or 0's, or the processing time extraordinarily long, increasing σ^2 could help.

11. POETDiscretePhase's computation time (inclusively disk IO) (compiled under Linux with `make opt`) for a 100 s NIRS signal invoked by `./poet_discrete_phase -ifn RawNirsSignals/measurement1.csv -v 20000 -K 7 -cg0 0.97 -cgh 0.945 -ci 6 -dr -or 0.083 -N 10000` is ≈18.3 s on Intel®'s Core™2Duo @2.66 GHz and ≈38 s on Intel®'s Pentium® 4 @2.4 GHz.
12. POETDiscretePhase (command in 11) consumes ≈150 MB of memory.
13. Fig. 3.8 shows results of POETDiscretePhase. The grey curves in A&C were measured on an adult; the cray curves E,G,I were measured on a newborn. The black curves in A-D were computed using the command in 4 with `-K 15` in A&B and `-K 5` in C&D. The black curves in E-J were computed similarly, but with `-cg0 0.94`, `-cgh 0.94`, `-K 10` in E&F, `-K 1` in G&H, `-K 2` in I&J. In A&E, the noise level is low, whereas in G quantisation noise is prominent. I features a prominent breathing component with period length ≈2 s distinct particularly in newborns. This component is here captured only through $\hat{\boldsymbol{A}}_{0,-}$; estimating the breathing component separately and then subtract it from the raw NIRS signal before estimating the heartbeat component might be advisable. C features a slight movement artefact between 5 s and 10 s.
14. Using the sum-product rule in the phase graph makes the phase estimates generally smoother compared to the max-product rule, especially with increasing number of iterations (option `-it`) in the "Coefficient Estimator"-"Phase Estimator" loop in Fig. 3.3. The number of iterations does not greatly matter when the max-product rule is used.

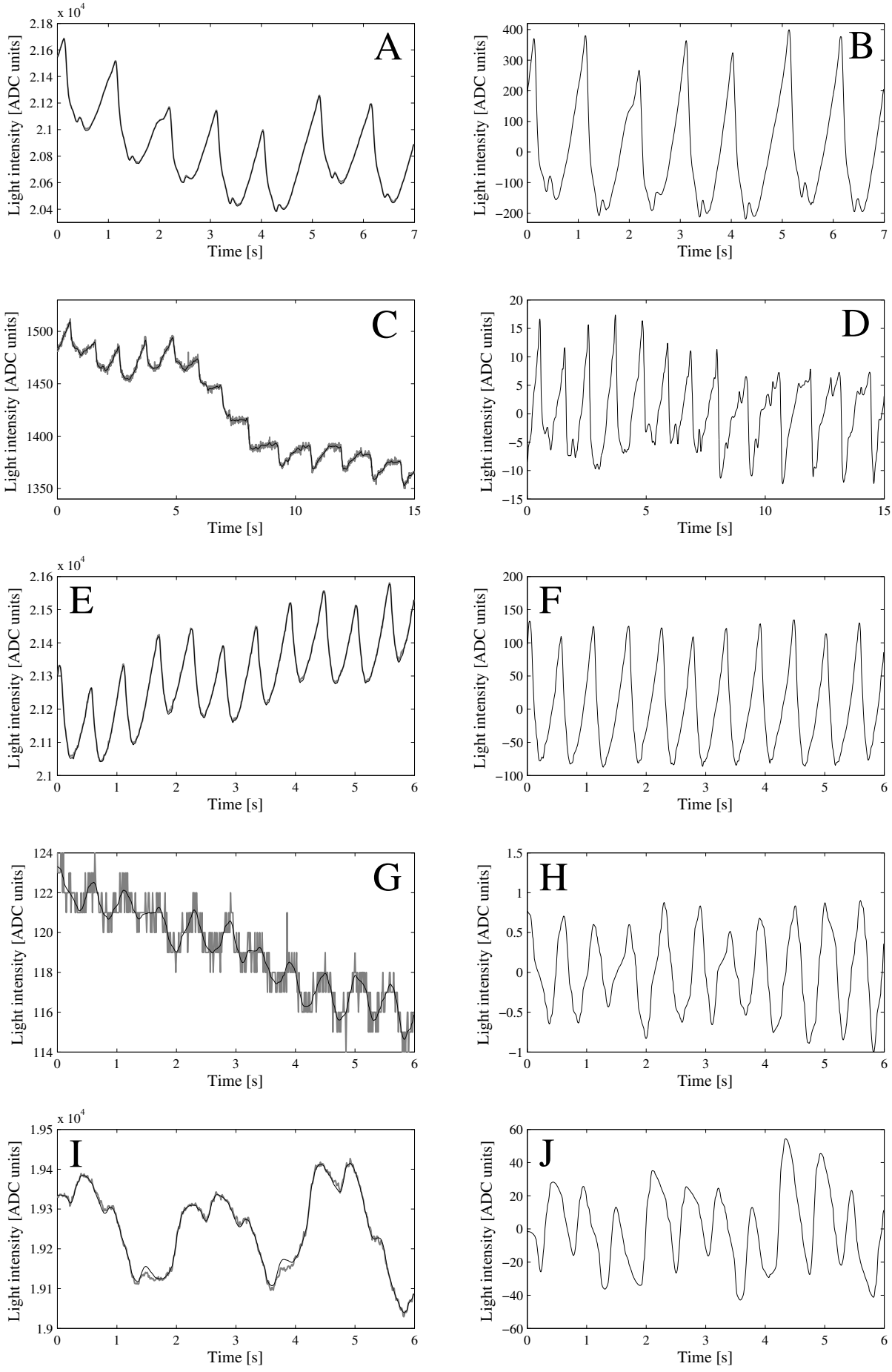

Figure 3.8: Examples of raw NIRS signals (grey curves in A, C, E, G, I) with sampling rate 100 Hz. The corresponding reconstructed heartbeat oscillations including the slow trends $\hat{\boldsymbol{A}}_{0,-}$, both computed with POETDiscretePhase, are the black curves in A, C, E, G, I; the reconstructed heartbeat oscillations without $\hat{\boldsymbol{A}}_{0,-}$ are in B, D, F, H, J. The signal values are given in "ADC units" since the NIRS instrumentation uses an ADC to digitise light intensity.

3.3 Application and validation

Title of the publication
Estimating and Validating the Interbeat Intervals of the Heart Using Near-Infrared Spectroscopy on the Human Forehead

Authors and affilations
Ivo Trajkovic [1,2], Felix Scholkmann [2], and Martin Wolf [2]

[1] Institute for Biomedical Engineering, University of Zurich and ETH Zurich, 8093 Zurich, Switzerland, trajkovic@biomed.ee.ethz.ch
[2] Biomedical Optics Research Laboratory, Division of Neonatology, Department of Obstetrics and Gynaecology, University Hospital Zurich, 8091 Zurich, Switzerland, {ivo.trajkovic, felix.scholkmann, martin.wolf}@usz.ch

Status
Submitted to the peer-reviewed *Journal of Biomedical Optics*, accepted for publication

Abstract
In studies with near-infrared spectroscopy, the recorded signals contain information on the temporal interbeat intervals of the heart. If exclusively this cardiac information is needed and could directly be extracted, an additional electrocardiography device would be unnecessary.

The aim was to estimate these intervals from signals measured with near-infrared spectroscopy with two novel approaches.

In one approach, we model the heartbeat oscillations in these signals with a Fourier series where the coefficients and the fundamental frequency can continuously change over time. The time-dependent model parameters are estimated and used to calculate the interbeat intervals.

The second approach uses empirical mode decomposition. The signal measured with near-infrared spectroscopy is empirically decomposed into a set of oscillatory components. The sum of a specific subset of them is an estimate of the pure heartbeat signal in which the diastolic peaks and consequential interbeat intervals are detected.

We show in simultaneous electrocardiography and near-infrared spectroscopy measurements on eleven subjects (8 men and 3 woman with mean age 32.8 ± 8.1 *years), that the interbeat intervals (and the consequential pulse rate variability measures), estimated using the proposed approaches, are in high agreement with their correspondents from electrocardiography.*

3.3.1 Introduction

Near-infrared spectroscopy (NIRS) potentially measures changes in oxy- and deoxyhemoglobin [19] caused by Mayer waves, breathing, cardiac activity and brain activity. Fig. 3.11A shows a NIRS signal featuring a heartbeat with a period of ≈1 s and a Mayer wave with a period of ≈10 s. The challenge is to extract the pure heartbeat and to estimate the heart rate variability (HRV) from NIRS signals.

HRV is (i) quantified by a set of statistical measures which result from time domain or frequency domain analysis of the time intervals between adjacent single heartbeats, called NN intervals, in an ECG measurement, (ii) a quantitative and diagnostic marker of the autonomic nervous system's control on the heart rate, (iii) used in research and clinical studies [26], e.g. a relationship between the autonomic nervous system's activity and cardiovascular mortality [27–29] has been shown.

The pulse rate variability (PRV) refers to the same set of statistical measures like HRV, but is extracted from photoplethysmography (PPG) signals. Since the operating principles of NIRS and PPG are similar, we use the term PRV when the measures are derived from NIRS signals; the term HRV is associated to ECG signals.

In NIRS signals, the heartbeat component is often present irrespective of the sensor's position. Thus, e.g. during functional studies, information on the NN intervals, and consequently on PRV, is already recorded. If exclusively this cardiac information is needed and would directly be extracted from NIRS, an ECG device would not be necessary.

In this paper, we focus on (i) describing two approaches how to estimate NN intervals from NIRS signals and (ii) showing proof of concept which is a basis for future clinical studies. The approaches are validated by comparing (i) their estimates with the corresponding ones from ECG and (ii) the resulting PRV and HRV measures, i.e. the standard deviation of the NN intervals (SDNN)

$$S = \sqrt{\frac{1}{L}\sum_{l=1}^{L}(\chi_l - \overline{\chi})^2} \tag{3.22}$$

with $\overline{\chi} = \frac{1}{L}\sum_{l=1}^{L}\chi_l$ and the square root of the mean squared differences of successive NN intervals (RMSSD)

$$R = \sqrt{\frac{1}{L-1}\sum_{l=1}^{L-1}(\chi_{l+1} - \chi_l)^2} \tag{3.23}$$

obtained from a set of L NN intervals $\chi_1, \ldots, \chi_L$. We omit frequency domain parameters, since they would not give additional insights concerning the agreement of NN intervals, SDNN and RMSSD between NIRS and ECG.

One approach uses Empirical Mode Decomposition (EMD); the other uses Parameter Estimation of a Model for Almost Periodic Signals (PEMAPS);

both algorithms were designed to analyse non-stationary signals which do not necessarily contain strictly periodic components.

3.3.2 Measurement

We tested the agreement between the NN intervals (and the resulting HRV measures) derived from ECG and the corresponding NN intervals (and the resulting PRV measures) estimated from NIRS. In 11 adult volunteers (8 men and 3 woman with mean age 32.8±8.1 years), ECG and NIRS were coregistered. During the experiment, each subject stood calmly, then sat calmly, and finally moved slowly both arms and hands while sitting; each of these three conditions took 5 minutes.

3.3.3 Instrumentation

The ECG device was a MK3-ETA made by TOM Medical Entwicklungs GmbH; the NIRS device was a continuous wave MCPII [4].

According to the user manual of the ECG device: electrode 1 was placed on the upper onset of the breastbone (sternal), electrode 2 was placed on the right lateral costal arch, and electrode 3 was placed submammary on the left.

The raw ECG signal is the sampled voltage difference between the electrodes 1 and 3. Recommended ECG sampling rates are 250 – 500 Hz [26]; we chose $f_{\text{ECG}} = 128$ Hz to make ECG signals and NIRS signals, the latter sampled at a not alterable rate of 100 Hz, comparable.

NN intervals were extracted by detecting the peaks of the R waves in a detrended, but not further filtered, ECG signal. Detrending made peak detection more robust; we considered further denoising unnecessary, since the R waves in ECG signals are strong compared to the noise.

Electrode 2 was used for potential equalisation.

The NIRS sensor is depicted in Fig. 3.9. Each source (light-emitting diode)

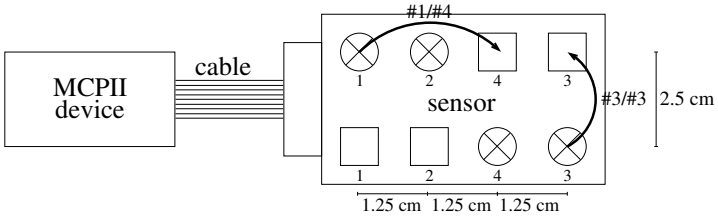

Figure 3.9: The NIRS sensor provides 4 light sources (circles) and 4 detectors (squares) and thus 16 light paths of which some are depicted as curved arrows.

sends light of constant intensity with wavelengths 750 nm, 800 nm and 875 nm; each detector (photodiode) measures light intensity. The NIRS signal's sample values are proportional to (i) the number of photons per time unit flowing through the photodiode, as well as (ii) the integral over the spectral sensitivity of the photodiode. MCPII was configured to drive 12 source/detector combinations, called "light paths", each with 3 wavelengths, resulting in 36 data channels. The sampling rate was $f_{\text{NIRS}} = 100$ Hz per data channel meaning that every 10 ms, 36 samples (1 sample per data channel) were acquired according to a time-multiplexed pattern.

NN intervals were estimated from one data channel which features a heartbeat component; the remaining 35 channels were ignored. The used light paths and wavelengths are stated in Table 3.1. The source/detector distances can be extracted from Fig. 3.9.

The NIRS sensor was placed on the right (from the subject's point of view) forehead where usually in several data channels a clear heartbeat component is present.

3.3.4 The approach based on PEMAPS

A periodic signal can be represented by a Fourier series. Specifically, one of the equidistant samples $x_1, x_2, \ldots$ of a real-valued periodic signal can be written as

$$x_n = \mathrm{Re}\left(\sum_{k=0}^{\infty} A_k e^{jkn\Omega}\right) \tag{3.24}$$

with real coefficient A_0, complex coefficients $A_1, A_2, \ldots$, and fundamental frequency $\Omega \in \mathbb{R}$.

We model the heartbeat component in NIRS signals as a trendless, almost periodic signal [30], i.e. $A_1, A_2, \ldots$ and Ω in Eq. (3.24) become time-dependent, and A_0 is discarded. Under this assumption, Eq. (3.24) changes to

$$x_n = \mathrm{Re}\left(\sum_{k=1}^{K} A_{k,n} \cdot e^{j\Theta_n k}\right) \tag{3.25}$$

with a single sample x_n of the real-valued heartbeat component at discrete time n, time-dependent coefficients $A_{1,n}, \ldots A_{K,n} \in \mathbb{C}$, time-dependent phase $\Theta_n \in [0, 2\pi]$, finite number of frequencies K, and

$$A_{k,n+1} \approx A_{k,n}, \tag{3.26}$$

$$\Theta_{n+1} = (\Theta_n + \Omega_n) \bmod 2\pi, \tag{3.27}$$

$$\Omega_{n+1} \approx \Omega_n. \tag{3.28}$$

Eq. (3.28) expresses the varying heart rate; Eq. (3.26) expresses the varying beat shape.

Let the NIRS signal be a noisy, trended version of the heartbeat component, i.e.

$$y_n = A_{0,n} + x_n + Z_n \tag{3.29}$$

which means a single NIRS sample y_n is a sum of the white Gaussian noise (sample Z_n), the heartbeat (sample x_n) and a slow trend (sample $A_{0,n}$). The latter models components slower than the heartbeat, i.e. Mayer waves, breathing and brain activity.

Given a vector of N measured NIRS samples $\boldsymbol{y} \triangleq (y_1, \ldots, y_N)$, the objective is to estimate the model parameter vector $\boldsymbol{\Theta} \triangleq (\Theta_1, \ldots, \Theta_N)$ and coefficient matrix

$$\mathbf{A} = \begin{pmatrix} A_{0,1} & \ldots & A_{0,N} \\ \vdots & \ddots & \vdots \\ A_{K,1} & \ldots & A_{K,N} \end{pmatrix}$$

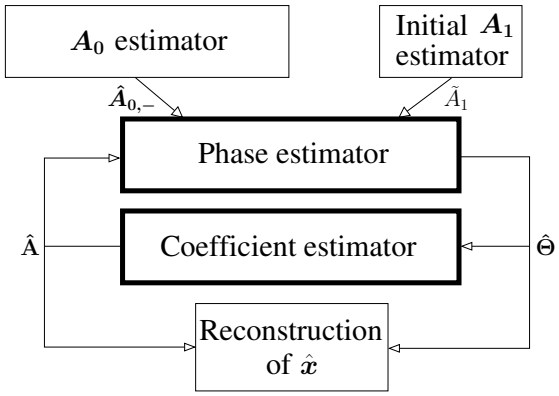

Figure 3.10: The building blocks of PEMAPS.

such that $\sum_{n=1}^{N}(y_n - x_n - A_{0,n})^2$ is minimal and reconstruct $\boldsymbol{x} \triangleq (x_1, \ldots, x_N)$ by applying the estimates in Eq. (3.25). We will use $\boldsymbol{A}_{k,-}$ for the k-th row (harmonic index) of $\mathbf{A}$.

Based on the measured NIRS samples $\boldsymbol{y}$ and a given estimate $\hat{\mathbf{A}}$ (we mark estimates of parameters with a hat, e.g $\hat{\mathbf{A}}$ is an estimate of $\mathbf{A}$), $\hat{\boldsymbol{\Theta}}$ is estimated as

$$\hat{\boldsymbol{\Theta}} = \underset{\boldsymbol{\Theta} \in [0,2\pi]^N}{\arg\max}\ f(\boldsymbol{\Theta} \mid \boldsymbol{y}, \hat{\mathbf{A}}) \tag{3.30}$$

where the conditional probability density function f in Eq. (3.30) comprises the assumption in Eq. (3.29), the model (3.25), the relation (3.27), and the constraint (3.28). The latter is handled with adjustable strength by using prior knowledge of the upper and lower limits of Ω_n. A heart rate has minimum $H_{\min}$ and maximum $H_{\max}$ values. Considering that $\boldsymbol{y}$ is sampled at f_{NIRS} [Hz] and that during one single heartbeat the phases $\Theta_1, \Theta_2, \ldots$ in Eq. (3.25) traverse the interval $[0, 2\pi]$, each heart rate H can be assigned to an angle growth Ω according to

$$\Omega(H) = \frac{H \cdot 2\pi}{f_{\text{NIRS}} \cdot 60}.$$

The limits of Ω_n are $\Omega_{\min} = \Omega(H_{\min})$ and $\Omega_{\max} = \Omega(H_{\max})$.

$A_{k,n}$ are estimated based on the measured NIRS samples $\boldsymbol{y}$, estimates $\hat{\boldsymbol{A}}_{k-1,-}, \ldots, \hat{\boldsymbol{A}}_{0,-}$, and $\hat{\boldsymbol{\Theta}}$ from the previous iteration as

$$\hat{A}_{k,n} = \underset{A_{k,n} \in \mathbb{C}}{\arg\max}\ g(A_{k,n} \mid \boldsymbol{y}, \hat{\boldsymbol{A}}_{k-1,-}, \ldots, \hat{\boldsymbol{A}}_{0,-}, \hat{\boldsymbol{\Theta}}) \tag{3.31}$$

for increasing k. The function g in (3.31) comprises the assumption in (3.29), (3.25) and the constraint (3.26). The latter is handled with adjustable strength by message damping as described in [30], section 3.4 ("$\overset{\gamma}{=}$"-node).

The probability density functions f in (3.30) and g in (3.31) are derived by using factor graphs and message passing algorithms described in sections 3.3 and 3.4 in [30].

The whole estimation algorithm is split into several building blocks whose interaction is depicted in Fig. 3.10.

Initially, the "$\boldsymbol{A}_0$ estimator" independently estimates the slow component $\boldsymbol{A}_{0,-}$ by means of (3.31). Since for this $\hat{\boldsymbol{\Theta}}$ is not needed, it is a one-time procedure based on $\boldsymbol{y}$ only.

In the heartbeat component, most of the signal energy, apart from the noise, lies in the fundamental frequency coefficient $\boldsymbol{A}_{1,-}$; thus a first rough estimate of the heartbeat component is a sinusoid. Its magnitude $\tilde{A}_1$ is calculated by the "Initial A_1 estimator" block such that the sinusoid is of approximately the same energy as $\boldsymbol{y} - \hat{\boldsymbol{A}}_{0,-}$.

Based on $\hat{\boldsymbol{A}}_{0,-}$ and $\tilde{A}_1$, the "Phase estimator" calculates $\hat{\boldsymbol{\Theta}}$ which is used by the "Coefficient estimator" to calculate the full set of coefficient estimates $\hat{\mathbf{A}}$. For the specific algorithms of the "Phase estimator" and the "Coefficient estimator" refer to [30], sections 3.3 and 3.4.

At this point, it is possible to enter entries of $\hat{\boldsymbol{\Theta}}$ and $\hat{\mathbf{A}}$ directly in (3.25) (done in the "Reconstruction of $\hat{x}$"-block) and obtain a first estimate $\hat{\boldsymbol{x}}$ of the heartbeat component. The result can be improved by iterating in the "Coefficient Estimator"-"Phase Estimator" loop (Fig. 3.10) before calculating $\hat{\boldsymbol{x}}$.

Fig. 3.11 illustrates how NN intervals are estimated from NIRS signals using PEMAPS. To convert the intervals Q in plot E to units of time, i.e. [s],

$$t_Q = \frac{Q}{f_s} \tag{3.32}$$

with sampling rate f_s [Hz] can be used.

3.3.5 The approach based on EMD

EMD decomposes a signal into a finite number of oscillatory modes, called Intrinsic Mode Functions (IMFs), by their characteristic time scales. The IMFs are derived empirically from the measured signal without any prior knowledge or model.

In [15], section 5) an IMF is formally defined, and the decomposition procedure, called "the sifting process", is described in detail. The latter can be summarised as follows:

1. Set $i = 1$.
2. Set $\boldsymbol{u} \triangleq \boldsymbol{y}$, with measured samples $\boldsymbol{y} = y_1, y_2, \ldots$.
3. Set the initial residual $\boldsymbol{r}_1 \triangleq \boldsymbol{y}$ (relevant in step 8).
4. Find the local extrema of $\boldsymbol{u}$.
5. Fit a cubic spline through all maxima which is an upper envelope $\boldsymbol{e}_u$ of $\boldsymbol{u}$, analogously through all minima which is a lower envelope $\boldsymbol{e}_l$ of $\boldsymbol{u}$.
6. The mean of the two envelopes $\boldsymbol{m}_i = (\boldsymbol{e}_l + \boldsymbol{e}_u)/2$ is subtracted from $\boldsymbol{u}$: $\boldsymbol{h}_i = \boldsymbol{u} - \boldsymbol{m}_i$.
7. Treat $\boldsymbol{h}_i$ as the new input signal $\boldsymbol{u}$ and go to step 4. Steps 4 - 7. are repeated until $\boldsymbol{h}_i$ becomes a curve with (i) as many zero crossings as extrema and (ii) the upper and lower envelopes of $\boldsymbol{h}_i$ become symmetric with regard to the zero line. Then $\boldsymbol{h}_i$ is the i-th IMF denoted as $\boldsymbol{c}_i$.
8. Let $\boldsymbol{u} \triangleq \boldsymbol{r}_{i+1} = \boldsymbol{r}_i - \boldsymbol{c}_i$.

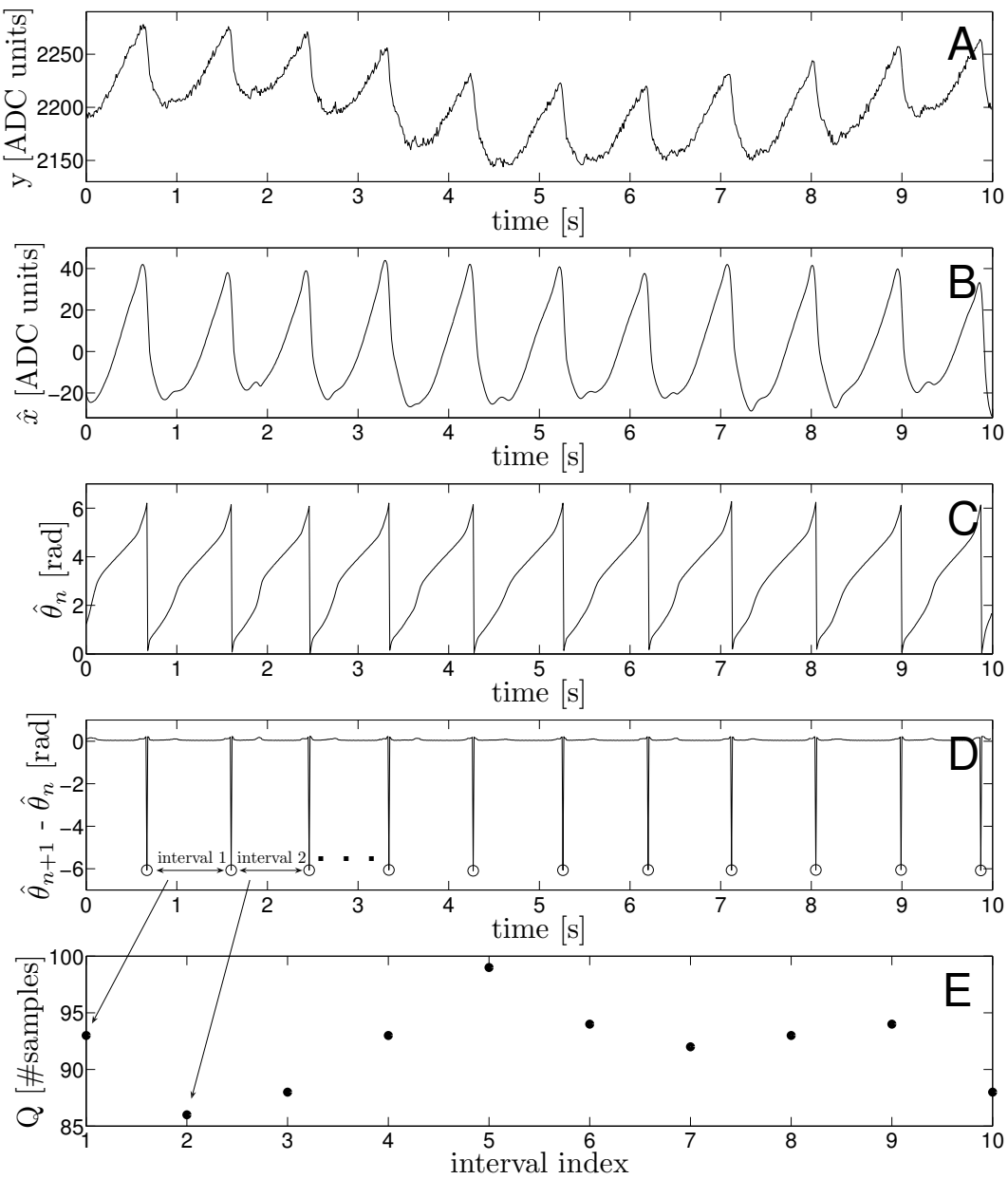

Figure 3.11: A shows a raw NIRS signal. B shows the heartbeat component resulting from applying $\hat{\boldsymbol{A}}_{1,-}, \ldots, \hat{\boldsymbol{A}}_{K,-}$ and $\hat{\boldsymbol{\Theta}} = (\hat{\Theta}_1, \ldots, \hat{\Theta}_{1000})$ in (3.25). C shows $\hat{\boldsymbol{\Theta}}$. D depicts $\hat{\Theta}'_1, \ldots, \hat{\Theta}'_{999}$ with $\hat{\Theta}'_n = \hat{\Theta}_{n+1} - \hat{\Theta}_n$. Since with increasing n, the values $\hat{\Theta}_n$ increase monotonically with respect to the modulo 2π function (dictated by (3.27)), $\hat{\Theta}'_n$ must be positive for all n except for the transition indices where the modulo operator takes effect (peaks in D). Finally, the samples between adjacent peaks are counted (E).

9. Increase i and go to step 4.

The sifting process stops when $\boldsymbol{r}_{i+1}$ in step 8 is a constant, a monotonic slope or a function with only one extremum. The original signal is given as

$$\boldsymbol{y} = \sum_{i=1}^{N} \boldsymbol{c}_i + \boldsymbol{r}_{N+1}.$$

The IMFs with lower indices i represent fast and those with higher indices slow oscillations.

By examining the IMFs by eye, one can recognise (based on their mean period length) which ones of them are related to the heartbeat component. The sum

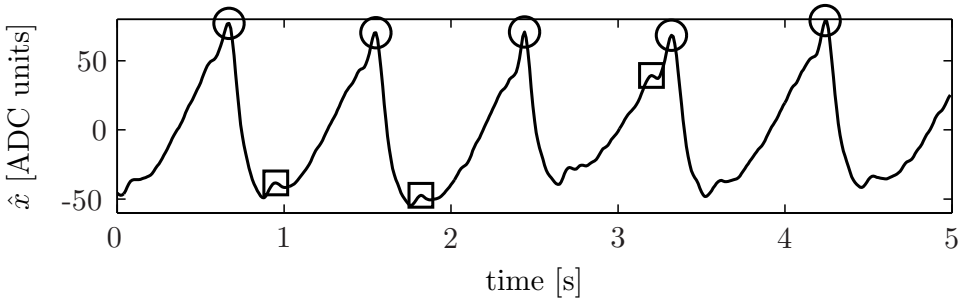

Figure 3.12: An estimate of the heartbeat component computed using EMD. All diastolic maxima are marked with circles, some of the non-diastolic maxima are marked with squares.

of this subset of IMFs represents an estimate $\hat{\boldsymbol{x}}$ of the heartbeat component of which an example is illustrated in Fig. 3.12.

In NIRS, near-infrared light penetrates tissue. The more blood the tissue contains, the more light is absorbed, i.e. the less light reaches the detector. Consequently, the light intensity at the detector is highest during diastole. We use the diastolic maxima (circles in Fig. 3.12) to estimate the NN intervals.

$\hat{x}_n$ is assumed to be the m-th diastolic maximum P_m with entry index $I(P_m) = n$ in $\hat{\boldsymbol{x}}$, if it is the m-th entry of $\hat{\boldsymbol{x}}$ for which the conditions

$$\begin{aligned} \hat{x}_n - \hat{x}_{n-1} &> 0, \\ \hat{x}_n - \hat{x}_{n+1} &> 0, \\ \hat{x}_n &> \xi_1, \qquad (3.33) \\ I(P_m) - I(P_{m-1}) &> \xi_2 \qquad (3.34) \end{aligned}$$

hold. With appropriate ξ_1 (e.g. $\xi_1 \triangleq 45$ in Fig. 3.12), the condition (3.33) discards the non-diastolic maxima (marked by squares in Fig. 3.12). Sometimes a diastolic maximum is lower than a non-diastolic one; then there is no ξ_1 such that the diastolic maximum is detected and the non-diastolic one is omitted. Choosing ξ_1 low enough such that all diastolic (and as well some non-diastolic) maxima are detected, using (3.34) to find neighbouring maxima which are too close to each other, and discarding the smaller maximum solves the problem.

The NN intervals can now be estimated by counting the samples between the detected adjacent diastolic maxima. Finally, (3.32) can be used to convert the intervals to units of time.

3.3.6 Validation: Data analysis

All recorded NIRS and ECG signals were evaluated offline. Segments with movement artefacts (sudden, typically between 0.5 s and 2 s long changes during which the standard deviation of the signal increases more than 100%) were excluded from the evaluation. In all our signals, such changes are distinct, and we recognised them easily by eye. Alternatively, the algorithm in [31] can be used to detect excessive values in the standard deviation of a NIRS or ECG signal which are larger than an intuitively chosen threshold (see [31], section 2.2.2). The corresponding NIRS and ECG samples are then assumed to be distorted by movement artefacts. Since in our case the latter are distinct without edge cases, this algorithm would exclude the same segments as we recognised by eye.

One subject was excluded due to technical problems.

From NIRS, NN intervals were estimated from a data channel with a clear heartbeat component; the remaining data channels were ignored.

From a detrended ECG signal, NN intervals were estimated by detecting the peaks of the R waves with the same procedure as detecting diastolic peaks in the heartbeat component estimated using EMD (procedure described at the end of Section 3.3.5). For every subject, ξ_1 in (3.33) was chosen individually. For all subjects, $\xi_2 = 61$ samples in (3.34).

For all subjects, PEMAPS was set up with $K = 3$ in (3.25), two iterations in the "Phase estimator"/"Coefficient estimator" loop (Fig. 3.10), message damping factors for harmonic indices $k = 0$: $\gamma = 0.936$, $k = 1$: $\gamma = 0.98$, $k = 2$: $\gamma = 0.985$ and $k = 3$: $\gamma = 0.989$ (see [30], section 3.4, "$\overset{\gamma}{=}$"-node) and $\Omega_{\min} = 0.026$ rad. and $\Omega_{\max} = 0.131$ rad (see Section 3.3.4). $\hat{\boldsymbol{A}}_{0,-}$ was smoothed by feeding it back to the "$\boldsymbol{A}_0$ estimator" (Fig. 3.10) as the new input signal and reestimating $\boldsymbol{A}_{0,-}$. For every subject this procedure was performed twice successively.

For the EMD-based approach, ξ_1 was chosen for every subject individually; $\xi_2 = 48$ samples.

3.3.7 Validation: Results

Assuming that, for the j-th subject we evaluated a set of W_j NN intervals $\boldsymbol{\chi}^{(j)}_{\text{ECG}} \triangleq (\chi^{(j)}_{\text{E},1}, \ldots, \chi^{(j)}_{\text{E},W_j})$ derived from ECG and $\boldsymbol{\chi}^{(j)}_{\text{NIRS}} \triangleq (\chi^{(j)}_{\text{N},1}, \ldots, \chi^{(j)}_{\text{N},W_j})$ estimated from NIRS using PEMAPS and EMD; Table 3.1 shows, for each subject, i.e. for $j = 1, \ldots, 11$, the cross-correlation coefficient

$$r \triangleq \frac{1}{W_j - 1} \frac{\sum_{w=1}^{W_j} (\chi^{(j)}_{\text{E},w} - \overline{\chi}^{(j)}_{\text{ECG}}) \cdot (\chi^{(j)}_{\text{N},w} - \overline{\chi}^{(j)}_{\text{NIRS}})}{\hat{\sigma}^{(j)}_{\chi_{\text{E}}} \cdot \hat{\sigma}^{(j)}_{\chi_{\text{N}}}} \tag{3.35}$$

with empirical means $\overline{\chi}^{(j)}_{\text{ECG}} = \frac{1}{W_j} \sum_{w=1}^{W_j} \chi^{(j)}_{\text{E},w}$ and $\overline{\chi}^{(j)}_{\text{NIRS}} = \frac{1}{W_j} \sum_{w=1}^{W_j} \chi^{(j)}_{\text{N},w}$, and empirical standard deviations $\hat{\sigma}^{(j)}_{\chi_{\text{E}}} = \sqrt{\frac{1}{W_j - 1} \sum_{w=1}^{W_j} (\chi^{(j)}_{\text{E},w} - \overline{\chi}^{(j)}_{\text{ECG}})^2}$ and $\hat{\sigma}^{(j)}_{\chi_{\text{N}}} = \sqrt{\frac{1}{W_j - 1} \sum_{w=1}^{W_j} (\chi^{(j)}_{\text{N},w} - \overline{\chi}^{(j)}_{\text{NIRS}})^2}$. In addition, for each subject, (i) the used wavelength, (ii) light path and (iii) an estimate of the signal-to-noise ratio

$$\text{SNR} = \frac{\sum_{n=1}^{N} \hat{x}_n^2}{\sum_{n=1}^{N} (y_n - \hat{x}_n - \hat{A}_{0,n})^2} \tag{3.36}$$

with reconstructed heartbeat sample $\hat{x}_n$ in (3.25), NIRS sample y_n and estimated sample $\hat{A}_{0,n}$ of the slow trend in (3.29) are given in Table 3.1.

Table 3.2 shows, per subject, how SDNN changes (i) from standing to sitting and (ii) during sitting from resting to moving hands. Table 3.3 shows the same for RMSSD. In both tables, (i) the physiological adaptation phases, i.e. the first 40 s during sitting and the first 10 s during the exercise, were excluded from the analysis, and (ii) SDNN / RMSSD values of all 11 subjects are compared between ECG and NIRS using cross-correlation coefficients r and p-values (t-test); the usability of the t-test has been approved by a Lilliefors test.

Subj.	r ECG/PEMAPS	r ECG/EMD	SNR	Wave-len.[nm]	Light-path
1	0.992376	0.981996	60.5	750	#1/#1
2	0.996571	0.996401	76.4	750	#1/#1
3	0.997600	0.995015	104.4	750	#1/#1
4	0.998201	0.997232	100.8	750	#1/#1
5	0.991168	0.981781	56.2	875	#4/#1
6	0.996770	0.988709	126.9	800	#4/#1
7	0.993744	0.984243	101.3	800	#4/#1
8	0.988616	0.969616	205.6	750	#1/#1
9	0.997752	0.996418	131.7	750	#1/#1
10	0.993719	0.990270	83.2	750	#1/#1
11	0.995355	0.987103	51.4	750	#1/#4
median	0.995355	0.988709	100.8	-	-

Table 3.1: Cross correlation coefficients of NN intervals as defined in Eq. (3.35)

Subj.	From standing to sitting			From resting to exercise		
	ECG	PEMAPS	EMD	ECG	PEMAPS	EMD
1	-5.16	-3.10	-0.09	17.83	15.09	12.79
2	9.97	8.87	8.38	-29.71	-28.63	-28.44
3	-25.00	-23.10	-22.14	-0.24	-0.12	-0.90
4	40.73	40.67	39.98	-47.16	-47.40	-47.11
5	-10.02	-9.67	-9.31	15.24	13.71	12.73
6	33.59	31.70	31.10	-38.96	-39.32	-39.27
7	-22.25	-21.49	-21.53	-14.16	-13.95	-13.27
8	-10.68	-10.32	-9.66	15.50	15.53	11.53
9	-18.29	-17.71	-17.47	1.07	1.70	0.89
10	-28.38	-28.15	-26.85	30.07	30.39	29.00
11	22.49	21.30	23.02	-20.28	-18.11	-19.03
r		0.9993	0.9975		0.9989	0.9979
p		$4.61 \cdot 10^{-14}$	$1.22 \cdot 10^{-11}$		$3.46 \cdot 10^{-13}$	$5.1 \cdot 10^{-12}$

Table 3.2: Change of SDNN (in %) related to condition changing; cross correlation coefficients r and p-values (t-test) give a comparison of all 11 SDNN values between ECG and NIRS

Let $\boldsymbol{\chi}_{\text{ECG}}$ be a vector of NN intervals derived from ECG of all 11 subjects; let $\boldsymbol{\chi}_{\text{PEMAPS}}$ and $\boldsymbol{\chi}_{\text{EMD}}$ be the same, but estimated from NIRS using PEMAPS and EMD. The values of $\boldsymbol{\chi}_{\text{ECG}}$, $\boldsymbol{\chi}_{\text{PEMAPS}}$ and $\boldsymbol{\chi}_{\text{EMD}}$ at a given index are 3 estimates of the same NN interval. Fig. 3.13 shows the agreement between $\boldsymbol{\chi}_{\text{ECG}}$ and $\boldsymbol{\chi}_{\text{PEMAPS}}$ on the one hand, $\boldsymbol{\chi}_{\text{ECG}}$ and $\boldsymbol{\chi}_{\text{EMD}}$ on the other hand. In all plots in this section, the lack of agreement between two measures is summarised by the mean $\hat{\mu}$ and the standard deviation $\hat{\sigma}$ of their difference points.

Due to the sampling, the entries of $\boldsymbol{\chi}_{\text{ECG}}$, $\boldsymbol{\chi}_{\text{PEMAPS}}$, $\boldsymbol{\chi}_{\text{EMD}}$, and thus the differences $\boldsymbol{\chi}_{\text{ECG}}$ - $\boldsymbol{\chi}_{\text{PEMAPS}}$ and $\boldsymbol{\chi}_{\text{ECG}}$ - $\boldsymbol{\chi}_{\text{EMD}}$, are discrete. Hence, a regular raster can be recognised in Fig. 3.13 and often the pair of values at a given index in $\boldsymbol{\chi}_{\text{ECG}}$ and $\boldsymbol{\chi}_{\text{PEMAPS}}$ or $\boldsymbol{\chi}_{\text{ECG}}$ and $\boldsymbol{\chi}_{\text{EMD}}$ is not unique. Thus, the size of a single point

Subj.	From standing to sitting			From resting to exercise		
	ECG	PEMAPS	EMD	ECG	PEMAPS	EMD
1	76.76	67.47	64.43	-22.24	-17.45	-11.74
2	79.27	74.42	73.73	-42.98	-40.00	-40.33
3	7.52	6.80	11.60	-21.48	-18.27	-21.74
4	70.65	65.84	64.90	-51.40	-51.16	-49.50
5	45.85	45.71	28.17	-21.47	-19.45	-14.50
6	51.37	48.28	47.89	-30.92	-31.64	-32.07
7	30.00	29.65	7.70	-28.30	-24.76	-18.19
8	-2.68	-5.06	2.92	7.97	20.10	-4.62
9	6.10	9.42	8.72	-47.63	-40.71	-41.72
10	10.26	6.85	12.53	-6.51	-0.70	-7.44
11	32.63	29.70	28.07	-30.53	-23.66	-23.81
r		0.9963	0.9561		0.9892	0.9289
p		$6.37 \cdot 10^{-11}$	$4.28 \cdot 10^{-06}$		$8.14 \cdot 10^{-09}$	$3.59 \cdot 10^{-05}$

Table 3.3: Change of RMSSD (in %) related to condition changing; cross correlation coefficients r and p-values (t-test) give a comparison of all 11 RMSSD values between ECG and NIRS

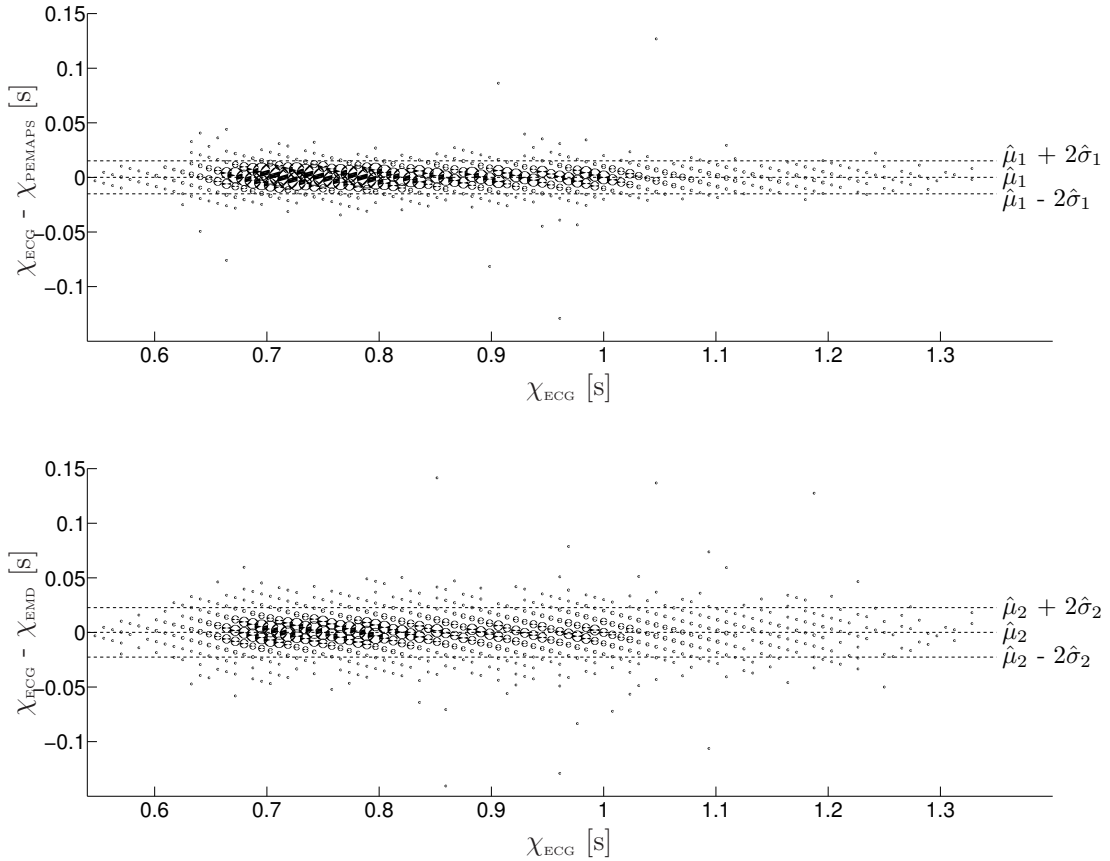

Figure 3.13: Agreement between NN intervals in all subjects derived from ECG and those estimated from NIRS using PEMAPS (upper plot), and using EMD (lower plot); $\hat{\mu}_1 = 0.00008$ s, $\hat{\sigma}_1 = 0.00755$ s, $\hat{\mu}_2 = 0.00008$ s and $\hat{\sigma}_2 = 0.01133$ s. This type of plots in conjunction with testing agreement was proposed in [32]. The size of a single point in the plot is proportional to the number of occurrences of the corresponding entry pairs.

in Fig. 3.13 is proportional to the number of occurrences of the corresponding entry pairs.

The mean of NN intervals in all subjects (derived from ECG signals), i.e.

the mean of x-coordinates of all points in Fig. 3.13, is 0.84 s. A NN interval estimated with PEMAPS or EMD differs from this mean on average by $\frac{\hat{\sigma}_1}{0.84\text{ s}} = 0.9\%$, or $\frac{\hat{\sigma}_2}{0.84\text{ s}} = 1.35\%$ respectively.

Fig. 3.14 shows SDNN, calculated from ECG using (3.22), S_{ECG}, versus its difference to S_{PEMAPS} and S_{EMD}. Fig. 3.15 shows the same for RMSSD. The mean of

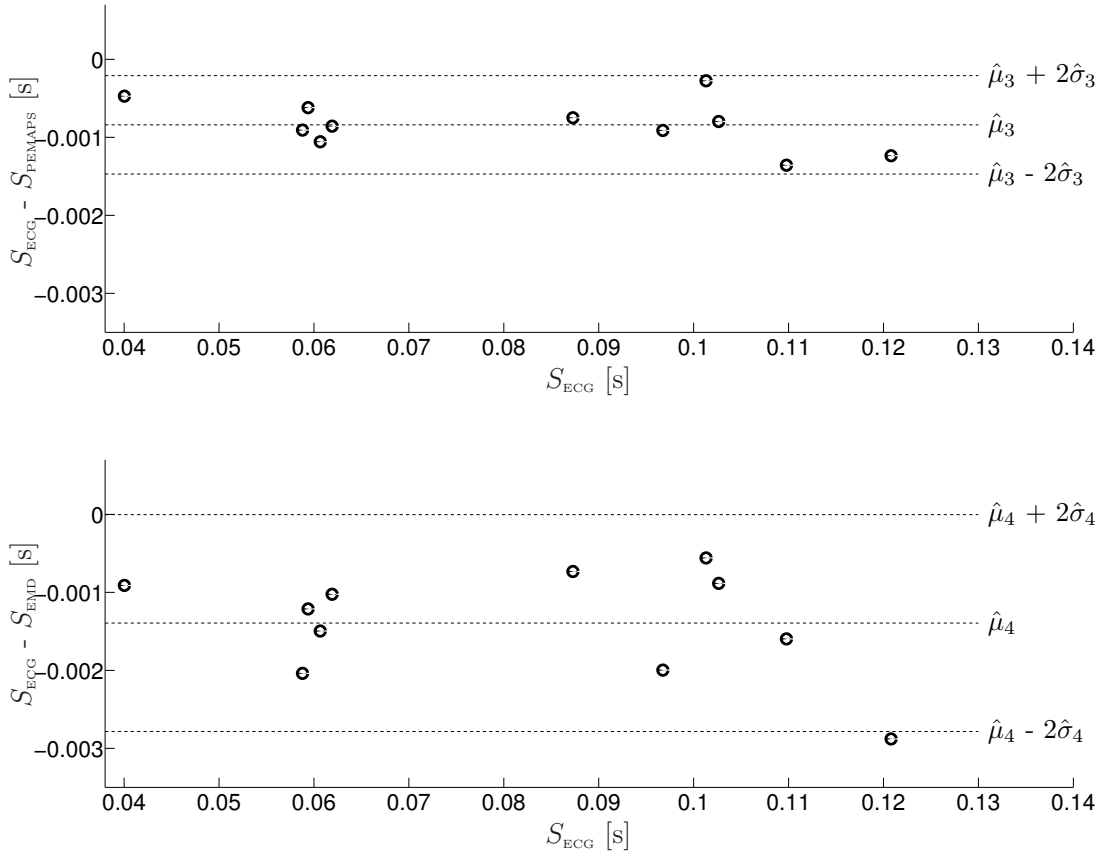

Figure 3.14: Agreement between SDNN derived from ECG and SDNN estimated from NIRS using PEMAPS (upper plot), and EMD (lower plot); $\hat{\mu}_3 = -0.00084$ s, $\hat{\sigma}_3 = 0.00032$ s, $\hat{\mu}_4 = -0.0014$ s and $\hat{\sigma}_4 = 0.0007$ s. Every point was derived from the entire experiment with one subject.

SDNN values in all subjects (derived from ECG), i.e. the mean of x-coordinates of all points in Fig. 3.14, is 0.08s. A SDNN value, derived with PEMAPS or EMD, differs from this mean on average by $\frac{\hat{\sigma}_3}{0.08\text{ s}} = 0.4\%$ or $\frac{\hat{\sigma}_4}{0.08\text{ s}} = 0.88\%$ respectively. The average differences for RMSSD are $\frac{\hat{\sigma}_5}{0.038\text{ s}} = 2.55\%$ and $\frac{\hat{\sigma}_6}{0.038\text{ s}} = 5.16\%$.

In Fig. 3.14 and Fig. 3.15, the differences between the two measures are biased, i.e. $\hat{\mu}_3, \ldots, \hat{\mu}_6$ are different from zero. This is caused by an error when detecting R peaks (ECG) and diastoles (NIRS) in discrete-time signals. The issue is explained in the appendix of this paper.

3.3.8 Discussion and Conclusion

As stated at the end of Section 3.3.7, the average discrepancies between SDNN from ECG and SDNN from NIRS are 0.4% (PEMAPS) and 0.88% (EMD). In [27], the risk of mortality was compared between two groups of subjects; in one group SDNN < 0.05 s, in the second group SDNN > 0.1 s which is >100% higher than in the first group. Conclusively, SDNN derived from NIRS

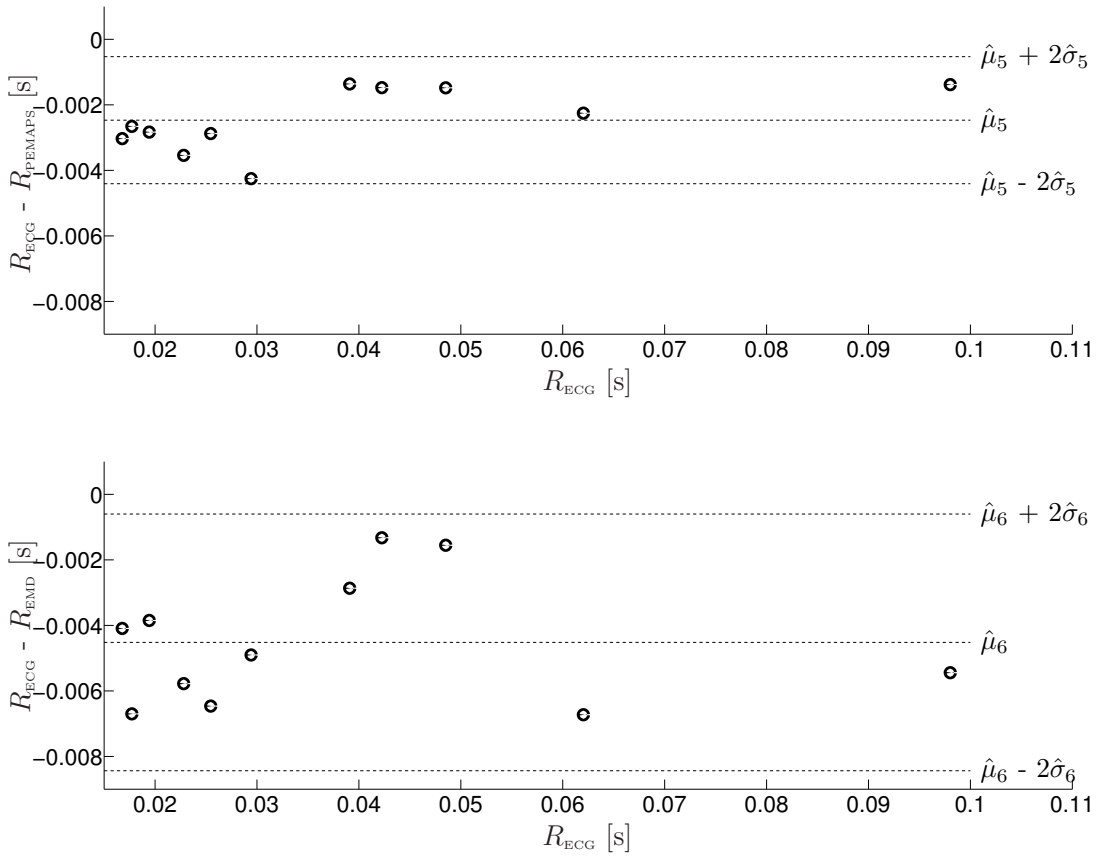

Figure 3.15: Agreement between RMSSD derived from ECG and RMSSD estimated from NIRS using PEMAPS (upper plot), and EMD (lower plot); $\hat{\mu}_5 = -0.00247$ s, $\hat{\sigma}_5 = 0.00097$ s, $\hat{\mu}_6 = -0.00452$ s and $\hat{\sigma}_6 = 0.00196$ s. Every point was derived from the entire experiment with one subject.

with the proposed approaches could be sufficiently accurate to derive the risk of mortality.

As shown in [15], EMD can be applied to signals from various sources in nature and does not depend on amplitude or frequency of the oscillations to be reconstructed. The version of PEMAPS used in this work is limited to signals with a strong (initial estimate $\tilde{A}_1$, Section 3.3.4) and rather high ($\gtrapprox$ 0.4 Hz) fundamental frequency.

The NIRS instrumentation should capture at least the fundamental frequency of the heartbeat oscillation, i.e. the sampling rate of NIRS should at least be $2H_{\max}$=6 Hz if $H_{\max} = 180$ bpm is the maximal heart rate. We tested this successfully by interpolating a 6 Hz NIRS signal to 100 Hz from which the NN intervals were derived with the proposed approaches. The correlation coefficients were as high as deriving the NN intervals from a real 100 Hz NIRS signal. Interpolating was necessary, since low sampling rates induce an error when deriving NN intervals. In [33], for example, the influence of this error on the power spectrum of the NN intervals is quantified. We address the influence of this error on SDNN and RMSSD in the appendix of this paper.

Both methods can be used with arbitrary signal lengths. The only limiting factor is RAM. If the amount of RAM does not suffice to analyse a 24 h recording as one block, the signals can be split into several blocks, each being analysed separately. The implementations we used in this work require for a

15 min. long NIRS signal $\approx$ 1.3 GB (PEMAPS) and $\approx$ 60 MB (EMD) of RAM.

Between subjects, the used wavelengths and SNR (3.36) vary considerably which has no high impact on the correlation coefficients in Table 3.1, e.g. compare subjects 1 and 8 concerning SNR and subjects 5 and 11 concerning wavelength.

The higher SDNN, the larger the variability between all NN intervals in the evaluated NIRS or ECG signal. The higher RMSSD, the larger the variability between successive NN intervals in the evaluated NIRS or ECG signal. According to [34], RMSSD relates mainly to the parasympathethic activity, whereas SDNN relates to sympathetic and parasympathetic activity. In all subjects (except for subject 8) in Table 3.3, RMSSD from ECG considerably (i) increases from standing to sitting and (ii) decreases during sitting from resting to performing the exercise which is also detectable from NIRS signals using PEMAPS and EMD. On the contrary, SDNN shows no such uniform changes over all subjects. In each subject in Table 3.2, the changes of SDNN from NIRS agree with their ECG correspondent more than the changes of RMSSD. In addition, the more the RMSSD and the SDNN values differ from 0, the better they agree between ECG and NIRS.

In [35–38], NN intervals and HRV measures derived from ECG were compared to their correspondents from PPG. In [35], correlations $1 > r > 0.97$ were found between HRV and PRV in 44 subjects. Frequency domain and time domain measures were calculated. All signals were sampled at 400 Hz. In [36], where 10 subjects participated, these correlations are in the range $0.99985 > r > 0.962$. All signals were sampled at 400 Hz. In [38], the median correlation coefficient of the NN intervals derived from ECG and their PPG correspondents in 10 subjects is $r = 0.97$ (compared to the medians 0.995355 and 0.988709 in Table 3.1). All signals were sampled at 1 kHz. In [37], the same coefficient was derived from 42 subjects as $r = 0.91$ (including an outlier). The ECG signals were sampled at 200 Hz; PPG signals were sampled at 100 Hz. We conclude that our results show slightly higher agreement although we used considerably lower sampling rates.

Many factors can affect the agreement between NN intervals derived from ECG and the corresponding ones estimated from NIRS using PEMAPS and EMD, e.g. external forces on the arterial vessels, pathologies, methodical problems, small and unnoticed movement artefacts, and variability in time which the pulse pressure waveform takes to propagate through the arterial tree. Our results show, that with healthy subjects who (i) are not exposed to mechanical forces and (ii) behave calmly during the experiment, the impact of these factors is negligible.

PEMAPS estimates correspond closer to ECG than the EMD estimates, i.e. $\hat{\sigma}_1 < \hat{\sigma}_2$ in Fig. 3.13, $|\hat{\mu}_3| < |\hat{\mu}_4|$ and $\hat{\sigma}_3 < \hat{\sigma}_4$ in Fig. 3.14, and $|\hat{\mu}_5| < |\hat{\mu}_6|$ and $\hat{\sigma}_5 < \hat{\sigma}_6$ in Fig. 3.15.

Compared to EMD, the version of PEMAPS used in this work is (i) slower and requires more RAM, (ii) less error-prone, and (iii) set up in a more general way, i.e. the input arguments of the PEMAPS implementation are the same

for all subjects. When using EMD, the user must examine the intrinsic mode functions (see Section 3.3.5) by eye and set ξ_1 in (3.33) manually for every subject.

According to [1], the clinical outlook of near-infrared techniques is noninvasive (i) brain imaging by providing functional and metabolic maps of the activated brain cortex, (ii) measurement of changes and absolute values in oxy- and deoxyhemoglobin, (iii) measurement of blood pressure changes, and (iv) measurement of respiratory rate. In all these applications, NIRS coregisters information on NN intervals; this could give new physiological insights. Furthermore, in multimodal measurement setups with strong electromagnetic fields, e.g. as caused by magnetic resonance imaging or computed tomography, ECG may be disturbed while NIRS would function properly.

3.3.A Appendix: SDNN and sampling

In this section, based on the error induced by the sampling, (i) $\hat{\mu}_3 < 0$ and $\hat{\mu}_4 < 0$ in Fig. 3.14 are motivated, and (ii) a lower bound for $\hat{\sigma}_1$ and $\hat{\sigma}_2$ in Fig. 3.13 is derived.

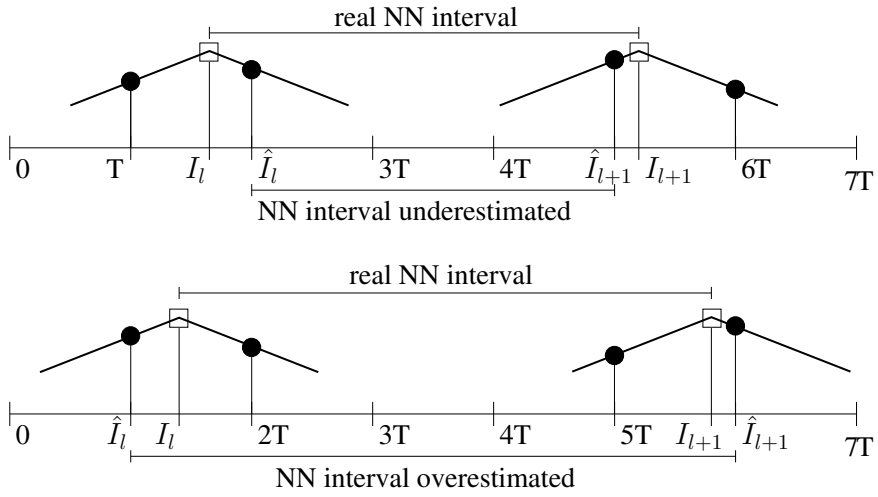

Figure 3.16: The angulated line pairs represent R waves in a continuous-time ECG signal which is sampled (circles) at times 0, T, 2T, ... The positions of the R wave peaks are unknown. Each R peak (marked with an unfilled square) is inclosed by two samples; the higher one is detected.

Let I_l be the unknown (continuous-time) position of the l-th maximum, and let $\hat{I}_l$ be the (discrete-time) position of the detected maximum (see Fig. 3.16). The continuous-valued discrepancy $D_l = I_l - \hat{I}_l$ is assumed to be uniformly distributed over the interval $[-\frac{T}{2}, \frac{T}{2}]$ with variance $\sigma_D^2 = \frac{T^2}{12}$.
The discrepancy between the l-th real and the l-th detected NN interval is given as

$$\begin{aligned} \Lambda_l &= I_{l+1} - I_l - (\hat{I}_{l+1} - \hat{I}_l) \\ &= D_{l+1} - D_l. \end{aligned} \tag{3.37}$$

It follows that the difference of two uniformly distributed random variables Λ_l must be triangularly distributed over the interval $[-T, T]$ with variance $\sigma_\Lambda^2 = 2\sigma_D^2 = \frac{T^2}{6}$. In our case, this variance is $\sigma_{\Lambda_\mathrm{E}}^2 = \frac{T_\mathrm{ECG}^2}{6} = \frac{1}{6 f_\mathrm{ECG}^2} \approx 0.00001017\ \mathrm{s}^2$

(ECG) and $\sigma^2_{\Lambda_\mathrm{N}} = \frac{1}{6 f^2_{\mathrm{NIRS}}} \approx 0.00001667\ \mathrm{s}^2$ (NIRS).
By rearranging (3.37), it follows that

$$\hat{I}_{l+1} - \hat{I}_l = I_{l+1} - I_l - \Lambda_l. \tag{3.38}$$

Let the l-th entry of $\boldsymbol{\chi}_{\mathrm{ECG}}$ in Fig. 3.13 be modelled, according to (3.38), as

$$\chi_{\mathrm{E},l} \triangleq I_{l+1} - I_l - \Lambda_{\mathrm{E},l} \tag{3.39}$$

with discrepancy $\Lambda_{\mathrm{E},l}$ between the l-th real and the l-th detected NN interval; analogously

$$\chi_{\mathrm{N},l} \triangleq I_{l+1} - I_l - \Lambda_{\mathrm{N},l} \tag{3.40}$$

in the case of NIRS. Let $\sigma^2_{\chi_\mathrm{E}}$ be the variance of $\chi_{\mathrm{E},l}$ and let σ^2_χ be the variance of $I_{l+1} - I_l$. With respect to (3.39),

$$\sigma^2_{\chi_\mathrm{E}} = \sigma^2_\chi + \sigma^2_{\Lambda_\mathrm{E}}. \tag{3.41}$$

Likewise, let $\sigma^2_{\chi_\mathrm{N}}$ be the variance of $\chi_{\mathrm{N},l}$. With respect to (3.40),

$$\sigma^2_{\chi_\mathrm{N}} = \sigma^2_\chi + \sigma^2_{\Lambda_\mathrm{N}}. \tag{3.42}$$

From $\sigma_{\Lambda_\mathrm{E}} < \sigma_{\Lambda_\mathrm{N}}$, (3.41) and (3.42), it follows that $\sigma_{\chi_\mathrm{E}} < \sigma_{\chi_\mathrm{N}}$; this motivates $\hat{\mu}_3 < 0$ and $\hat{\mu}_4 < 0$ in Fig. 3.14, since (i) S_{ECG} is an estimator of σ_{χ_E} and (ii) S_{EMD} and S_{PEMAPS} are estimators of σ_{χ_N}

In Fig. 3.13, according to (3.39) and (3.40), the quantity on the y-axes can be modelled as $\chi_{\mathrm{E},l} - \chi_{\mathrm{N},l} = \Lambda_{\mathrm{N},l} - \Lambda_{\mathrm{E},l}$. The distribution of $\chi_{\mathrm{E},l} - \chi_{\mathrm{N},l}$ has standard deviation $\sqrt{\sigma^2_{\Lambda_\mathrm{E}} + \sigma^2_{\Lambda_\mathrm{N}}} \approx 0.00518$ s. The latter may be seen as a lower bound for $\hat{\sigma}_1$ and $\hat{\sigma}_2$, since the error in estimating NN intervals is not only caused by the sampling.

3.3.B Appendix: RMSSD and sampling

In this section, based on the error induced by the sampling, $\hat{\mu}_5 < 0$ and $\hat{\mu}_6 < 0$ in Fig. 3.15 are motivated.

By expanding the squared term in (3.23), it follows that the expectation value

$$E\big[R^2\big] = 2\sigma^2 + 2\mu^2 - \frac{2}{L-1} \sum_{l=1}^{L-1} E\big[\chi_{l+1} \cdot \chi_l\big] \tag{3.43}$$

with variance σ^2 and mean μ of χ_l.

Let μ_{χ_E} be the mean of $\chi_{\mathrm{E},l}$ in (3.39); let μ_{χ_N} be the mean of $\chi_{\mathrm{N},l}$ in (3.40); let $\mu_\chi$and be the mean of $I_{n+1} - I_n$. Since the means of $\Lambda_{\mathrm{E},l}$ and $\Lambda_{\mathrm{N},l}$ are 0,

$$\mu_{\chi_\mathrm{E}} = \mu_{\chi_\mathrm{N}} = \mu_\chi. \tag{3.44}$$

After (i) replacing χ_l by $\chi_{\mathrm{E},l}$, and thus σ^2 and μ by $\sigma^2_{\chi_\mathrm{E}}$ and μ_{χ_E}, in (3.43) and (ii) using (3.41) and (3.44), the expectation value of the square of R_{ECG} in

Fig. 3.15 is given as

$$E\big[R_{\text{ECG}}^2\big] = 2(\sigma_\chi^2 + \sigma_{\Lambda_\text{E}}^2) + 2\mu_{\chi_\text{E}}^2 - \ldots \\ \frac{2}{L-1}\sum_{l=1}^{L-1} E\big[\chi_{\text{E},l+1} \cdot \chi_{\text{E},l}\big]. \tag{3.45}$$

By (i) substituting $\chi_{\text{E},l}$ and $\chi_{\text{E},l+1}$ in (3.45) according to (3.39), (ii) then expanding their product, (iii) assuming that $\Lambda_{\text{E},l}$ is independent of I_l and I_{l+1} and (iv) using $E[\Lambda_{\text{E},l}] = 0$, it follows that

$$E\big[R_{\text{ECG}}^2\big] = 2(\sigma_\chi^2 + \sigma_{\Lambda_\text{E}}^2) + 2\mu_{\chi_\text{E}}^2 - \ldots \\ \frac{2}{L-1}\sum_{l=1}^{L-1} E\big[(I_{l+2} - I_{l+1})(I_{l+1} - I_l)\big]. \tag{3.46}$$

Analogously, it follows from (3.42) and (3.44) that

$$E\big[R_{\text{NIRS}}^2\big] = 2(\sigma_\chi^2 + \sigma_{\Lambda_\text{N}}^2) + 2\mu_{\chi_\text{N}}^2 - \ldots \\ \frac{2}{L-1}\sum_{l=1}^{L-1} E\big[(I_{l+2} - I_{l+1})(I_{l+1} - I_l)\big]. \tag{3.47}$$

With respect to (3.44), the difference between (3.46) and (3.47) is

$$E\big[R_{\text{ECG}}^2\big] - E\big[R_{\text{NIRS}}^2\big] = 2\sigma_{\Lambda_\text{E}}^2 - 2\sigma_{\Lambda_\text{N}}^2.$$

From $\sigma_{\Lambda_\text{E}}^2 < \sigma_{\Lambda_\text{N}}^2$ (paragraph after (3.37)), it follows that

$$E\big[R_{\text{ECG}}^2\big] - E\big[R_{\text{NIRS}}^2\big] < 0$$

which motivates $\hat{\mu}_5 < 0$ and $\hat{\mu}_6 < 0$ in Fig. 3.15.

Acknowledgement

We thank all subjects for their cooperation and time, Patrick Flandrin and Gabriell Rilling for the fast EMD implementation (http://perso.ens-lyon.fr/patrick.flandrin/emd.html) we used in this work, Swiss National Science Foundation (SNF) for funding this project, Christoph Reller from the Signal and Information Processing Laboratory (ISI) at ETH Zurich for constructive discussions.

End of publication

Chapter 4

A faster approach to modelling and estimating almost periodic signals

4.1 The new phase estimator

Title of the publication

Modelling and Filtering of Physiological Oscillations in Near-Infrared Spectroscopy by Time-Varying Fourier Series

---

Authors and affilations

Ivo Trajkovic [1,2], Christoph Reller [3], and Martin Wolf [2]

[1] Institute for Biomedical Engineering, University of Zurich and ETH Zurich, 8093 Zurich, Switzerland, trajkovic@biomed.ee.ethz.ch

[2] Biomedical Optics Research Laboratory, Division of Neonatology, Department of Obstetrics and Gynaecology, University Hospital Zurich, 8091 Zurich, Switzerland, {ivo.trajkovic, martin.wolf}@usz.ch

[3] Signal and Information Processing Laboratory (ISI), ETH Zurich, 8092 Zurich, Switzerland, reller@isi.ee.ethz.ch

---

Status

Submitted to the peer-reviewed *Advances in Experimental Medicine and Biology*, accepted for publication.

Abstract

Raw near-infrared spectroscopy signals contain oscillatory components, namely low frequency oscillations (Mayer waves), breathing and the heartbeat. We propose an approach to model and estimate them from noisy measurements assuming that they are linearly superposed. Estimating them is important, since they pose disturbing effects, but they are also of scientific interest.

These components are not strictly periodic; we characterise them as "almost periodic". The model of an almost periodic signal is a Fourier series where the Fourier coefficients and the fundamental frequency are allowed to (slowly) change over time. This model can be represented by factor graphs which we use to derive message passing algorithms to estimate the time-dependent model parameters from a measured signal.

An implementation of the proposed algorithm processes a 100 s long measurement in 2 s (on a modern PC) which is ≈ 10 *times faster than a comparable previous implementation. Thus, real-time applications, e.g. online monitoring, could be realised using slower, inexpensive or power-saving hardware. The increase in speed was achieved by using a different parameterisation of the model*

which allows Gaussian message passing (with only 2 parameters: mean and variance), whereas previously some messages were digitised.

In the previous implementation, the number of harmonics in the model is chosen manually (for each subject and data channel). In this paper, we show an intuitive procedure to estimate this number from the measured signal. In conclusion, the proposed algorithm is able to separate the heartbeat and, in contrast to the previous implementation, the low frequency oscillation effectively and in real time.

4.1.1 Introduction

We have developed and implemented a method for modelling and adaptive filtering of oscillatory components, in particular the ones caused by (i) the cardiac activity, called heartbeat, and (ii) the low frequency oscillations (also Mayer waves), called LFO, in signals measured with near-infrared spectroscopy (NIRS), called raw NIRS signals. Examples of the latter are the grey curves in Fig. 4.3A and 4.3C. The overall aim is to extract each pure signal component, since (i) one disturbs the detection of another, e.g. hemodynamic brain activity and LFO, (ii) then the interrelation between several components can be assessed, and (iii) characterising each component separately could yield new understandings of underlying biological processes.

A traditional band-pass filter [39] will not do, since (i) to include the typical sharp peaks in the heartbeat (grey curve in Fig. 4.3C), the high cutoff frequency must be rather high; thus, high-frequency noise survives the filtering, (ii) the physiology fluctuates, e.g. the heart rate doubles quickly after starting a physical exercise, and window-based processing allows the harmonics of one or more components, e.g. heartbeat and fast neuronal activity, to spectrally overlap in a window.

Our method has not been tested yet with the breathing component, thus only the heartbeat and the LFO are addressed in this paper.

Strictly periodic signals can be efficiently represented by Fourier series. Let $x_1, x_2, \ldots$ be the equidistantly sampled version of a strictly periodic real-valued signal, let n be the discrete time index, and let k be the harmonic index. Then

$$x_n = \mathrm{Re}\left(\sum_{k=0}^{\infty} A_k e^{jkn\Omega}\right) \tag{4.1}$$

with coefficients $A_0 \in \mathbb{R}$, $A_1, A_2, \ldots \in \mathbb{C}$ and fundamental frequency $\Omega \in [0, 2\pi]$.

We classify the oscillatory components as "almost periodic", since their period lengths and signal shapes drift over time [30]. We propose to describe such a component by a "Fourier series" with time-variant fundamental frequency (related to the varying period length) and time-variant coefficients (related to the varying signal shape), i.e. we change (4.1) into

$$x_n = \mathrm{Re}\left(\sum_{k=1}^{K} A_{k,n} \cdot e^{j\Theta_n k}\right). \tag{4.2}$$

with

$$A_{k,n+1} \approx A_{k,n}$$
$$\Theta_{n+1} = (\Theta_n + \Omega_n) \bmod 2\pi, \tag{4.3}$$

and

$$\Omega_{n+1} \approx \Omega_n. \tag{4.4}$$

In this paper, we present an algorithm for estimating the model parameters Θ_n which is considerably faster than the algorithm in [30].

When modelling the heartbeat, the magnitude of the coefficients $A_{k,n}$ in (4.2) typically decays with increasing harmonic index k. Consequently, for $k \geq K$, the harmonics of the heartbeat cannot be distinguished from the noise. The noise energy in raw NIRS signals, and thus K, varies depending on the data channel, NIRS instrumentation and subject; typically $3 \leq K \leq 7$ whereas $K = 1$ suffices for modelling the LFO. In (4.2), K was chosen manually (for each subject and data channel); if K is chosen too high, the reconstructed heartbeat will contain noise; if K is chosen too low, high-frequency parts of the heartbeat will not be modelled. At the end of Section 4.1.2, we show an intuitive procedure for estimating K from the measured signal.

4.1.2 Estimating the model parameters

Let the raw NIRS signal $\boldsymbol{y} = (y_1, \ldots, y_N)$ be a noisy, trended version of the signal $\boldsymbol{x} = (x_1, \ldots, x_N)$ in (4.2) where N is the signal length. Specifically,

$$\boldsymbol{y} = \boldsymbol{A}_{0,-} + \boldsymbol{x} + \boldsymbol{Z} \tag{4.5}$$

where $\boldsymbol{Z} = (Z_1, \ldots, Z_N)$ is discrete time white Gaussian noise, and the trend $\boldsymbol{A}_{0,-} = (A_{0,1}, \ldots, A_{0,N})$ models changes slower than the heartbeat, or the LFO respectively, and thus is omitted in (4.2). We will use the vectors $\boldsymbol{A}_{k,-} = (A_{k,1}, \ldots, A_{k,N})$ for $k = 0, \ldots, K$ and decorate estimates with a hat (e.g. $\hat{C}$ is an estimate of C).

Given y, the objective is to estimate the phases $\boldsymbol{\Theta} = (\Theta_1, \ldots, \Theta_N)$, K and the coefficient vectors $\boldsymbol{A}_{0,-}, \ldots, \boldsymbol{A}_{K,-}$ such that $||\boldsymbol{y} - \hat{\boldsymbol{x}} - \hat{\boldsymbol{A}}_{0,-}||^2$ is minimal, where x is the reconstructed signal by applying the estimates in (4.2).

The estimation algorithm consists of several building blocks (see Fig. 4.1).

Initially, the "A_0 estimator" estimates the slow trend $\boldsymbol{A}_{0,-}$ by a one-time procedure similar to low pass filtering and based on y only.

In the heartbeat and the LFO, most of the energy, apart from the noise, lies in the fundamental frequency coefficient $\boldsymbol{A}_{1,-}$. Thus, a first rough estimate of such an oscillatory component is a complex sinusoid with constant complex magnitude. The "Initial A_1 estimator"-block makes an estimate $\tilde{A}_1$ of this magnitude such that the sinusoid has approximately the same energy as $\boldsymbol{y} - \hat{\boldsymbol{A}}_{0,-}$.

The "Phase estimator" calculates the final estimate $\hat{\boldsymbol{\Theta}}$ of $\boldsymbol{\Theta}$ based on estimates $\hat{\boldsymbol{A}}_{0,-}$, $\tilde{A}_1$ and and (4.2) with $K = 1$ parameterised as

$$x_n = \mathrm{Re}\left(\tilde{A}_1 \cdot e^{j\Theta_n}\right) = \mathrm{Re}(\tilde{A}_1)\cos(\Theta_n) - \mathrm{Im}(\tilde{A}_1)\sin(\Theta_n) = \hat{\boldsymbol{A}}_1 \cdot \boldsymbol{C}_n \tag{4.6}$$

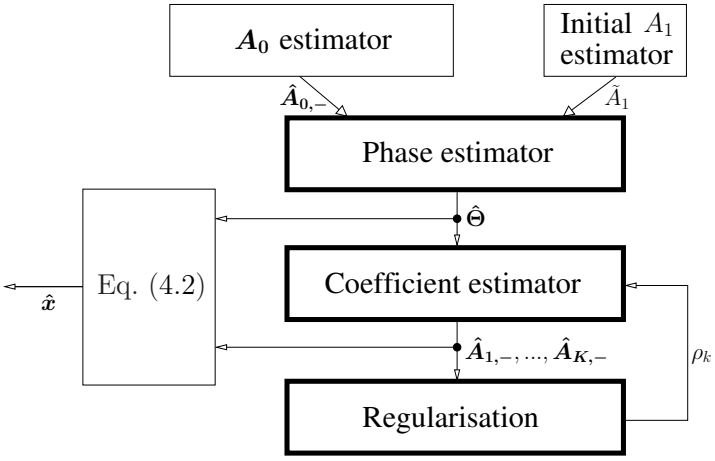

Figure 4.1: The building blocks of the proposed algorithm.

with constant vector $\hat{\boldsymbol{A}}_1 \triangleq \Big(\mathrm{Re}(\tilde{A}_1), \text{–}\mathrm{Im}(\tilde{A}_1)\Big)$. We introduce a state vector $\boldsymbol{C}_n \triangleq (\cos(\Theta_n), \sin(\Theta_n))^T$ and define a state transition

$$\boldsymbol{C}_n = \mathrm{rot}(\hat{\Omega}) \cdot \boldsymbol{C}_{n-1} + \boldsymbol{U}_n \tag{4.7}$$

where

$$\mathrm{rot}(\hat{\Omega}) = \begin{pmatrix} \cos(\hat{\Omega}) & -\sin(\hat{\Omega}) \\ \sin(\hat{\Omega}) & \cos(\hat{\Omega}) \end{pmatrix}$$

is a rotation matrix and $\hat{\Omega}$ is a prior estimate of Ω_n in (4.4). $\hat{\Omega}$ is derived by using the formula in [30], section 3.3, paragraph 4 and assuming a typical heart rate depending on the subject, e.g. $H \triangleq 80$ bpm for adults. Since $\hat{\Omega}$ is fixed, despite the fact that the heart rate varies considerably depending on various factors, uncertainty, i.e. two-dimensional zero-mean white Gaussian noise $\boldsymbol{U}_n$ with diagonal covariance matrix $\mathbf{V}$, is added to the rotated state in (4.7). This addition of noise defines (4.4). The frequency and its variability (and thus $\hat{\Omega}$ and $\mathbf{V}$) in the LFO are smaller than in the heartbeat.

The estimate $\hat{\boldsymbol{C}}_n$ of $\boldsymbol{C}_n$ is made as

$$\hat{\boldsymbol{C}}_n = \arg\max_{\boldsymbol{C}_n} f(\boldsymbol{C}_n \mid \hat{\boldsymbol{A}}_{0,-}, \tilde{A}_1, \boldsymbol{y}). \tag{4.8}$$

The function f in (4.8) comprises the equations (4.5), (4.6), and (4.7).

Each estimate $\hat{\Theta}_n$ in $\hat{\boldsymbol{\Theta}}$ is made as

$$\hat{\Theta}_n = \arctan \frac{\hat{\boldsymbol{C}}_n(2)}{\hat{\boldsymbol{C}}_n(1)} \tag{4.9}$$

with $\hat{\boldsymbol{C}}_n(i)$ denoting the i-th entry of the vector $\hat{\boldsymbol{C}}_n$.

In this paper, a node in the factor graph is either (i) the relationship between two or more variables, defined through an equation, or (ii) a prior probability density. Edges are variables.

The "Phase estimator" uses a factor graph containing N consecutive sections one of which is depicted in Fig. 4.2, i.e the outgoing edge $\boldsymbol{C}_{n+1}$ of the graph in the figure is at the same time the incoming edge of its right neighbour.

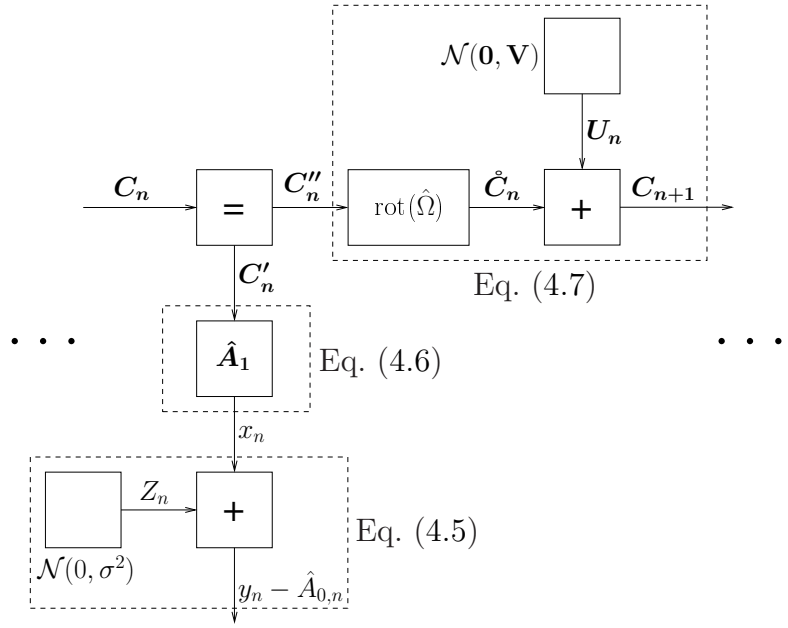

Figure 4.2: Factor graph used for estimating $\boldsymbol{C}_n$.

In Fig. 4.2, the "="-node clones $\boldsymbol{C}_n$: $\boldsymbol{C}'_n \triangleq \boldsymbol{C}_n$ and $\boldsymbol{C}''_n \triangleq \boldsymbol{C}_n$; the "$\hat{\boldsymbol{A}}_1$"-node represents (4.6); the section of the graph with the incoming edge x_n is (4.5); the section of the graph with the incoming edge $\boldsymbol{C}''_n$ is (4.7). The sum-product algorithm for Gaussian messages [21] is applied on the factor graph in Fig. 4.2 to derive f in (4.8).

A message (i) is a scaled conditional probability density of the underlying edge, (ii) can traverse the edge in both directions, and (iii) is named μ including an arrow, which indicates the forward ($\vec{\mu}$) or backward ($\overleftarrow{\mu}$) direction with respect to the edge direction, and the name of the underlying edge as a subscript (e.g. $\vec{\mu}_X$).

The schedule of the message passing algorithm on the factor graph in Fig. 4.2 is:

1. For $n = 1, \ldots, N$, calculate $\overleftarrow{\mu}_{x_n}$ from the measured sample y_n , estimate of the slow trend $A_{0,n}$ and the prior probability density $\vec{\mu}_{Z_n}$ represented by the "$\mathcal{N}(0, \sigma^2)$"-node.
2. For $n = 1, \ldots, N$, calculate $\overleftarrow{\mu}_{\boldsymbol{C}'_n}$ from $\overleftarrow{\mu}_{x_n}$ by means of [21], table 3.
3. For $n = 1, \ldots, N$, calculate in sequence (i) $\vec{\mu}_{\boldsymbol{C}''_n}$ from $\overleftarrow{\mu}_{\boldsymbol{C}'_n}$ and $\vec{\mu}_{\boldsymbol{C}_n}$ by means of [21], table 2 ($\vec{\mu}_{\boldsymbol{C}_1}$ is neutral: $\vec{\mu}_{\boldsymbol{C}_1} = 1$), (ii) $\vec{\mu}_{\mathring{\boldsymbol{C}}_n}$ from $\vec{\mu}_{\boldsymbol{C}''_n}$ by means of [21], table 3, since the rotation of the state $\boldsymbol{C}_n$ can be expressed as a matrix multiplication (4.7), (iii) $\vec{\mu}_{\boldsymbol{C}_{n+1}}$ from $\vec{\mu}_{\mathring{\boldsymbol{C}}_n}$ ([21], table 2).
4. For $n = N, \ldots, 1$, calculate in sequence (i) $\overleftarrow{\mu}_{\mathring{\boldsymbol{C}}_n}$ from $\overleftarrow{\mu}_{\boldsymbol{C}_{n+1}}$ ([21], table 2), (ii) $\overleftarrow{\mu}_{\boldsymbol{C}''_n}$ from $\overleftarrow{\mu}_{\mathring{\boldsymbol{C}}_n}$ ([21], table 3), and (iii) $\overleftarrow{\mu}_{\boldsymbol{C}_n}$ from $\overleftarrow{\mu}_{\boldsymbol{C}''_n}$ and $\overleftarrow{\mu}_{\boldsymbol{C}'_n}$ ([21], table 2). $\overleftarrow{\mu}_{\boldsymbol{C}_N}$ is neutral: $\overleftarrow{\mu}_{\boldsymbol{C}_N} = 1$.
5. For $n = 1, \ldots, N$, calculate in sequence (i) $\vec{\mu}_{\boldsymbol{C}'_n}$ from $\vec{\mu}_{\boldsymbol{C}_n}$ and $\overleftarrow{\mu}_{\boldsymbol{C}''_n}$, (ii) the marginal $\tilde{\mu}_{\boldsymbol{C}_n} = \vec{\mu}_{\boldsymbol{C}'_n} \cdot \overleftarrow{\mu}_{\boldsymbol{C}'_n}$ and (iii) the estimate $\hat{\Theta}_n$ according to (4.8) and (4.9) with $f \triangleq \tilde{\mu}_{\boldsymbol{C}_n}$ in (4.8).

The "Coefficient estimator" uses $\hat{\boldsymbol{\Theta}}$ to calculate the full set of coefficient estimates $\hat{\boldsymbol{A}}_{1,-}, \ldots, \hat{\boldsymbol{A}}_{K,-}$. The used factor graph and message passing algorithms

correspond to the ones in [30], section 3.4, with a slight modification.

To derive K in (4.2), in each iteration in the "Regularisation"-"Coefficient estimator" loop, the "Regularisation"-block calculates for $1 \leq k \leq K_{\max}$ the noise-to-coefficient ratio

$$\rho_k \triangleq \frac{||\boldsymbol{y} - \hat{\boldsymbol{x}}||^2}{||\hat{\boldsymbol{A}}_{k,-}||^2} \tag{4.10}$$

based on the estimate $\hat{\boldsymbol{A}}_{k,-}$ and the reconstructed signal $\boldsymbol{x}$ from the previous iteration. The higher ρ_k, the more the "Coefficient estimator" damps $\hat{\boldsymbol{A}}_{k,-}$ in the next iteration. This procedure requires slight modifications in the factor graph of the "Coefficient estimator" which are described in detail in [40]. The estimate of K is the largest $k \in [1, K_{\max}]$ for which $||\hat{\boldsymbol{A}}_{k,-}|| > \tau$, where τ is a threshold.

The "Eq. (4.2)"-block reconstructs the heartbeat, or the LFO respectively, by applying the estimates $\hat{\boldsymbol{\Theta}}$ and $\hat{\boldsymbol{A}}_{1,-}, \ldots, \hat{\boldsymbol{A}}_{K,-}$ in (4.2).

4.1.3 Results and Conclusions

The resulting (trended) heartbeat estimate (black curve in Fig. 4.3C) highly agrees with a corresponding estimate computed with the algorithm in [30] ($r = 0.999518$).

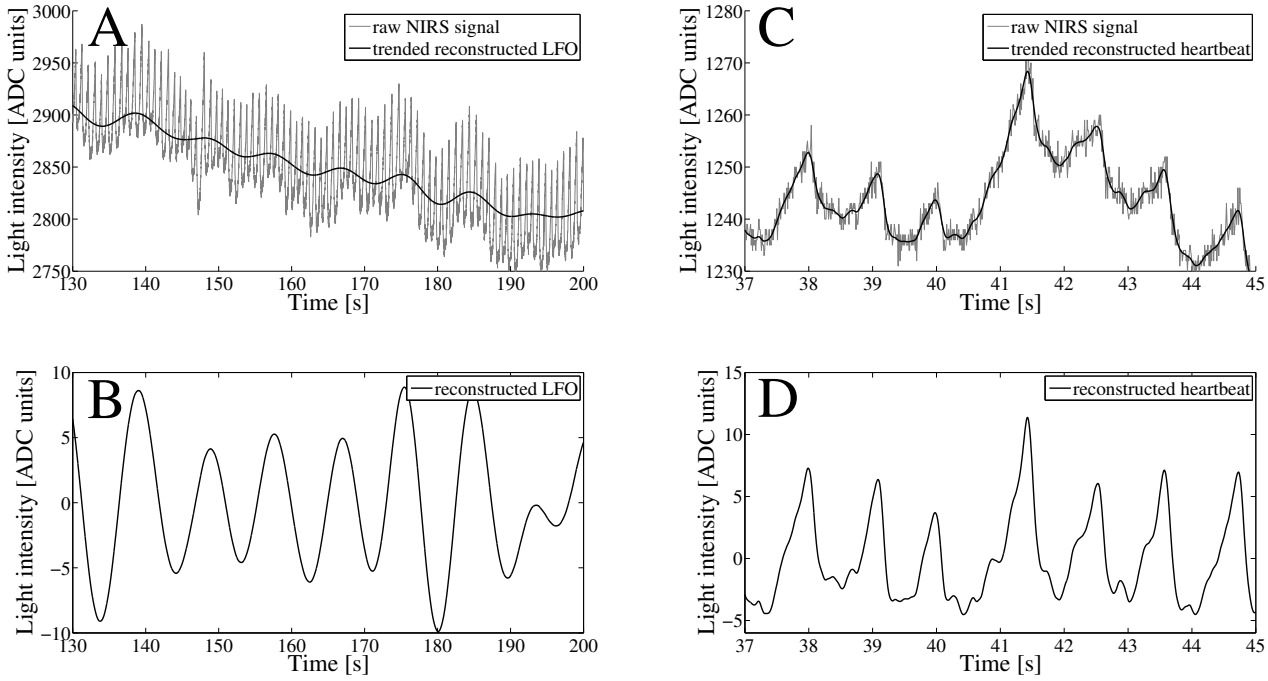

Figure 4.3: The grey curves in A and C are raw NIRS signals sampled at 100 Hz. The black curve in C is the reconstructed heartbeat including the estimated slow trend $\hat{\boldsymbol{A}}_{0,-}$; the black curve in A is the same for the LFO; B is the LFO without $\hat{\boldsymbol{A}}_{0,-}$; D is the heartbeat without $\hat{\boldsymbol{A}}_{0,-}$. The heartbeat is also recognisable in A (spikes in the grey curve). The signal values are given in ADC units since the NIRS instrumentation uses an ADC to digitise light intensity values.

Compared to the algorithm in [30], the algorithm proposed in this paper is (i) faster, (ii) able to estimate K from the measured signal, (iii) able to esti-

mate the LFO, and (iv) confirmed through many more results from functional studies [40].

End of publication

4.2 The new coefficient estimator

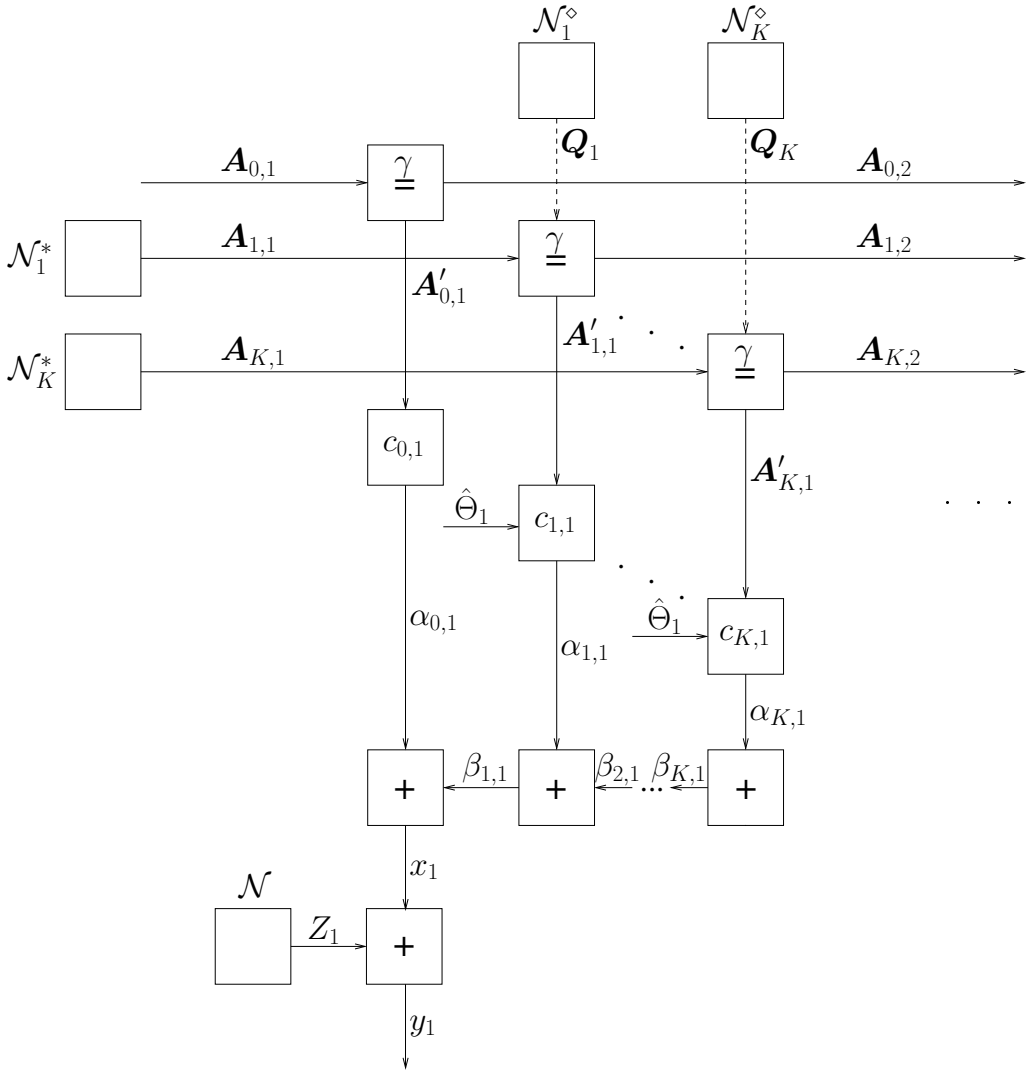

Figure 4.4: Factor graph used for estimating the coefficients $\boldsymbol{A}_{-,n}$ with regularisation.

The procedure for deriving K (see end of Section 4.1.2) in Eq. (4.2) uses "regularisation" in the factor graph in Fig. 4.4. Regularisation damps low-energy coefficients to prevent noise fitting. In particular, each marginal $\tilde{\mu}_{\boldsymbol{A}'_{k,n}}$ in step 6 in Section 3.1.7 is multiplied by a prior message $\vec{\mu}_{\boldsymbol{A}_{k,1}}$ with mean vector $\vec{\boldsymbol{m}}_{\boldsymbol{A}_{k,1}} \triangleq \mathbf{0}$ and covariance matrix $\vec{\mathbf{V}}_{\boldsymbol{A}_{k,1}} \triangleq \left(\begin{smallmatrix} \sigma_k^{*2} & 0 \\ 0 & \sigma_k^{*2} \end{smallmatrix}\right)$. A coefficient estimate is then made as

$$\hat{\boldsymbol{A}}_{k,n} = \arg\max_{\boldsymbol{A}_{k,n}} \tilde{\mu}_{\boldsymbol{A}'_{k,n}}(\boldsymbol{A}_{k,n}) \vec{\mu}_{\boldsymbol{A}_{k,1}}(\boldsymbol{A}_{k,n}) \tag{4.11}$$

$$= \left(\tilde{\mathbf{W}}_{\boldsymbol{A}_{k,n}} + \vec{\mathbf{V}}_{\boldsymbol{A}_{k,1}}{}^{-1}\right)^{-1} \tilde{\mathbf{W}}_{\boldsymbol{A}_{k,n}} \tilde{\boldsymbol{m}}_{\boldsymbol{A}_{k,n}} \tag{4.12}$$

$$= \begin{pmatrix} w_{11} + \frac{1}{\sigma_k^{*2}} & w_{12} \\ w_{21} & w_{22} + \frac{1}{\sigma_k^{*2}} \end{pmatrix}^{-1} \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{pmatrix} \tilde{\boldsymbol{m}}_{\boldsymbol{A}_{k,n}} \tag{4.13}$$

with $\tilde{\mathbf{W}}_{\boldsymbol{A}_{k,n}} = \left(\begin{smallmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{smallmatrix}\right)$ where all elements are positive. Eq. (4.12) follows from the facts that (i) the message computation rules for multiplying two messages coincide with the message computation rules in the "="-node (see [21], Section III.E), and (ii) a Gaussian PDF is maximal at its mean. In Eq. (4.13), increasing σ_k^{*2} makes the matrix product approaching an identity matrix, and the influence of regularisation on the coefficient estimate fades; decreasing σ_k^{*2} makes the matrix product approaching a zero matrix, and the coefficient estimate's energy decreases. Based on this, in each "Regularisation"-"Coefficient estimator" loop in Fig. 4.1, σ_k^{*2} is calculated according to

$$\sigma_k^{*2} = \frac{1}{\eta \rho_k} \tag{4.14}$$

with η adjusting the regularisation strength for all coefficients globally and noise-to-coefficient ratio ρ_k in Eq. (4.10).

In the following explanations, "forward pass" refers to calculating consecutively for $n = 2, \ldots, N$ the messages $\vec{\mu}_{\boldsymbol{A}_{k,n}}$; calculating $\overleftarrow{\mu}_{\boldsymbol{A}_{k,n}}$ for $n = N, \ldots, 1$ is called "backward pass"; calculating $\overleftarrow{\mu}_{\boldsymbol{A}'_{k,n}}$ for $n = 1, \ldots, N$ is called "upward pass"; calculating $\vec{\mu}_{\boldsymbol{A}'_{k,n}}$ for $n = 1, \ldots, N$ is called "downward pass". Furthermore, only the message parameters weight and weighted mean are considered, since the relevant computation rules (2.17)–(2.24) are based on them.

Computing the parameters of the message product in Eq. (4.11) is not done explicitly but is embedded in the message passing algorithm in the factor graph in Fig. 4.4 in which the node in Fig. 2.4 is used. The "$\mathcal{N}_k^*$"-node represents the prior $\vec{\mu}_{\boldsymbol{A}_{k,1}}$.

The upward pass in Fig. 4.4 is equivalent with the one in Fig. 3.5 (step 2 in Section 3.1.7).

The forward pass, with prior message $\vec{\mu}_{\boldsymbol{Q}_k} \triangleq (\vec{\mu}_{\boldsymbol{A}_{k,1}})^{1-\gamma}$ represented by the "$\mathcal{N}_k^\diamond$"-node, begins with computing $\vec{\mu}_{\boldsymbol{A}_{k,2}}$'s parameters according to Eq. (2.23) and (2.24) as

$$\begin{aligned}
\vec{\mathbf{W}}_{\boldsymbol{A}_{k,2}} &= \gamma \vec{\mathbf{W}}_{\boldsymbol{A}_{k,1}} + \overleftarrow{\mathbf{W}}_{\boldsymbol{A}'_{k,1}} + \vec{\mathbf{W}}_{\boldsymbol{Q}_k} \\
&= \gamma \vec{\mathbf{W}}_{\boldsymbol{A}_{k,1}} + \overleftarrow{\mathbf{W}}_{\boldsymbol{A}'_{k,1}} + (1-\gamma) \vec{\mathbf{W}}_{\boldsymbol{A}_{k,1}} \\
&= \vec{\mathbf{W}}_{\boldsymbol{A}_{k,1}} + \overleftarrow{\mathbf{W}}_{\boldsymbol{A}'_{k,1}} \qquad (4.15) \\
\vec{\mathbf{W}}_{\boldsymbol{A}_{k,2}} \vec{\boldsymbol{m}}_{\boldsymbol{A}_{k,2}} &= \gamma \vec{\mathbf{W}}_{\boldsymbol{A}_{k,1}} \vec{\boldsymbol{m}}_{\boldsymbol{A}_{k,1}} + \overleftarrow{\mathbf{W}}_{\boldsymbol{A}'_{k,1}} \overleftarrow{\boldsymbol{m}}_{\boldsymbol{A}'_{k,1}} + \vec{\mathbf{W}}_{\boldsymbol{Q}_k} \vec{\boldsymbol{m}}_{\boldsymbol{Q}_k} \\
&= \overleftarrow{\mathbf{W}}_{\boldsymbol{A}'_{k,1}} \overleftarrow{\boldsymbol{m}}_{\boldsymbol{A}'_{k,1}} \qquad (4.16)
\end{aligned}$$

Eq. (4.16) follows from $\vec{\boldsymbol{m}}_{\boldsymbol{Q}_k} \triangleq \mathbf{0}$ and $\vec{\boldsymbol{m}}_{\boldsymbol{A}_{k,1}} \triangleq \mathbf{0}$. By using Eq. (2.23) and Eq. (4.15), then Eq. (2.24) and Eq. (4.16), the forward pass proceeds with

$$\begin{aligned}
\vec{\mathbf{W}}_{\boldsymbol{A}_{k,3}} &= \gamma \vec{\mathbf{W}}_{\boldsymbol{A}_{k,2}} + \overleftarrow{\mathbf{W}}_{\boldsymbol{A}'_{k,2}} + \vec{\mathbf{W}}_{\boldsymbol{Q}_k}, \\
&= \gamma (\vec{\mathbf{W}}_{\boldsymbol{A}_{k,1}} + \overleftarrow{\mathbf{W}}_{\boldsymbol{A}'_{k,1}}) + \overleftarrow{\mathbf{W}}_{\boldsymbol{A}'_{k,2}} + (1-\gamma) \vec{\mathbf{W}}_{\boldsymbol{A}_{k,1}}, \\
&= \vec{\mathbf{W}}_{\boldsymbol{A}_{k,1}} + \gamma \overleftarrow{\mathbf{W}}_{\boldsymbol{A}'_{k,1}} + \overleftarrow{\mathbf{W}}_{\boldsymbol{A}'_{k,2}}. \qquad (4.17) \\
\vec{\mathbf{W}}_{\boldsymbol{A}_{k,3}} \vec{\boldsymbol{m}}_{\boldsymbol{A}_{k,3}} &= \gamma \overleftarrow{\mathbf{W}}_{\boldsymbol{A}'_{k,1}} \overleftarrow{\boldsymbol{m}}_{\boldsymbol{A}'_{k,1}} + \overleftarrow{\mathbf{W}}_{\boldsymbol{A}'_{k,2}} \overleftarrow{\boldsymbol{m}}_{\boldsymbol{A}'_{k,2}} \qquad (4.18)
\end{aligned}$$

and so forth with the parameters of $\vec{\mu}_{\boldsymbol{A}_{k,4}}, \vec{\mu}_{\boldsymbol{A}_{k,5}}, \ldots$. Eq. (4.15)–(4.18) show that defining $\vec{\mu}_{\boldsymbol{Q}_k} \triangleq (\vec{\mu}_{\boldsymbol{A}_{k,1}})^{1-\gamma}$ leaves $\vec{\mathbf{W}}_{\boldsymbol{A}_{k,1}}$ unaffected in each step in the forward

pass; in other words, $\vec{\mu}_{\boldsymbol{A}_{k,1}}$'s significance is not decreased even when this prior message is taken to the power of γ in each step in the forward pass. Thus, each $\vec{\mu}_{\boldsymbol{A}_{k,n}}$ for $n = 2, \ldots, N$ comprises the unaffected prior $\vec{\mu}_{\boldsymbol{A}_{k,1}}$, i.e. $\vec{\mu}_{\boldsymbol{A}_{k,n}}$ in Fig. 4.4 is equivalent with $\vec{\mu}_{\boldsymbol{A}_{k,n}}$ in Fig. 3.5 multiplied by $\vec{\mu}_{\boldsymbol{A}_{k,1}}$.

In the backward pass, the "$\mathcal{N}_k^*$"-node has no influence and thus the "$\mathcal{N}_k^\diamond$"-node is disabled or thought of as being neutral; then the message passing in Fig. 4.4 is equivalent with the one in Fig. 3.5.

The downward pass in Fig. 4.4 is equivalent with the one in Fig. 3.5 (step 5 in Section 3.1.7).

From the explanations above and with respect to Fig. 3.5, it follows that for $n = 1, \ldots, N$ each marginal $\tilde{\mu}_{\boldsymbol{A}'_{k,n}} = \vec{\mu}_{\boldsymbol{A}_{k,1}} \vec{\mu}_{\boldsymbol{A}'_{k,n}} \overleftarrow{\mu}_{\boldsymbol{A}'_{k,n}}$ (compare with step 6 in Section 3.1.7).

4.3 Implementation

This section describes the implementation of an enhanced algorithm. The latter uses the factor graph in Fig. 4.2 and the factor graph in Fig. 4.4. The implementation of this new algorithm is called POETGaussian, since exclusively Gaussian messages are used in both graphs.

The UML sequence diagram of POETGaussian is depicted in Fig. 4.5; the comments on the left establish a connection to Fig. 4.1 and the message passing algorithms in Sections 4.1.2 and 4.2.

The classes in POETGaussian are outlined below. The abbreviations are similar to those in Section 3.2. Since POETGaussian was derived from POETDiscretePhase, many classes, methods and variables are equivalent; their descriptions are omitted in this chapter, since they can be found in Chapter 3.

4.3.1 The contents of `Core.cpp`

The following objects and variables, defined in the global namespace in `Core.cpp`, should be pointed out.

- `SampleVector sample_woA0_buf` holds the detrended NIRS signal $\boldsymbol{y} - \hat{\boldsymbol{A}}_{0,-}$.
- `ParameterDataType reg_strength` holds the regularisation strength $0 < \eta < \infty$ in Eq. (4.14) (command line option `-rs`).
- `FGPhaseGaussian*` `fg_phase_gaussian` points to the object representing the phase graph.
- `FGCoeff*` `fg_coeff` points to the object representing the coefficient graph.
- `ParameterDataType U_var` holds σ_k^{*2} in Eq. (4.14) (command line option `-peuv`).
- `ParameterDataType omega_hat` holds $\hat{\Omega}$ in Eq. (4.7) (command line option `-oh`).

The following methods, defined in the global namespace in `Core.cpp`, should be pointed out.

- `void FGPhaseForwardPass(unsigned k)` calls `FGPhaseGaussian::GetLastMsgCForw(...)` (described in Section 4.3.3) with arguments (i) `msg_c_forw_first` referencing a neutral message (see

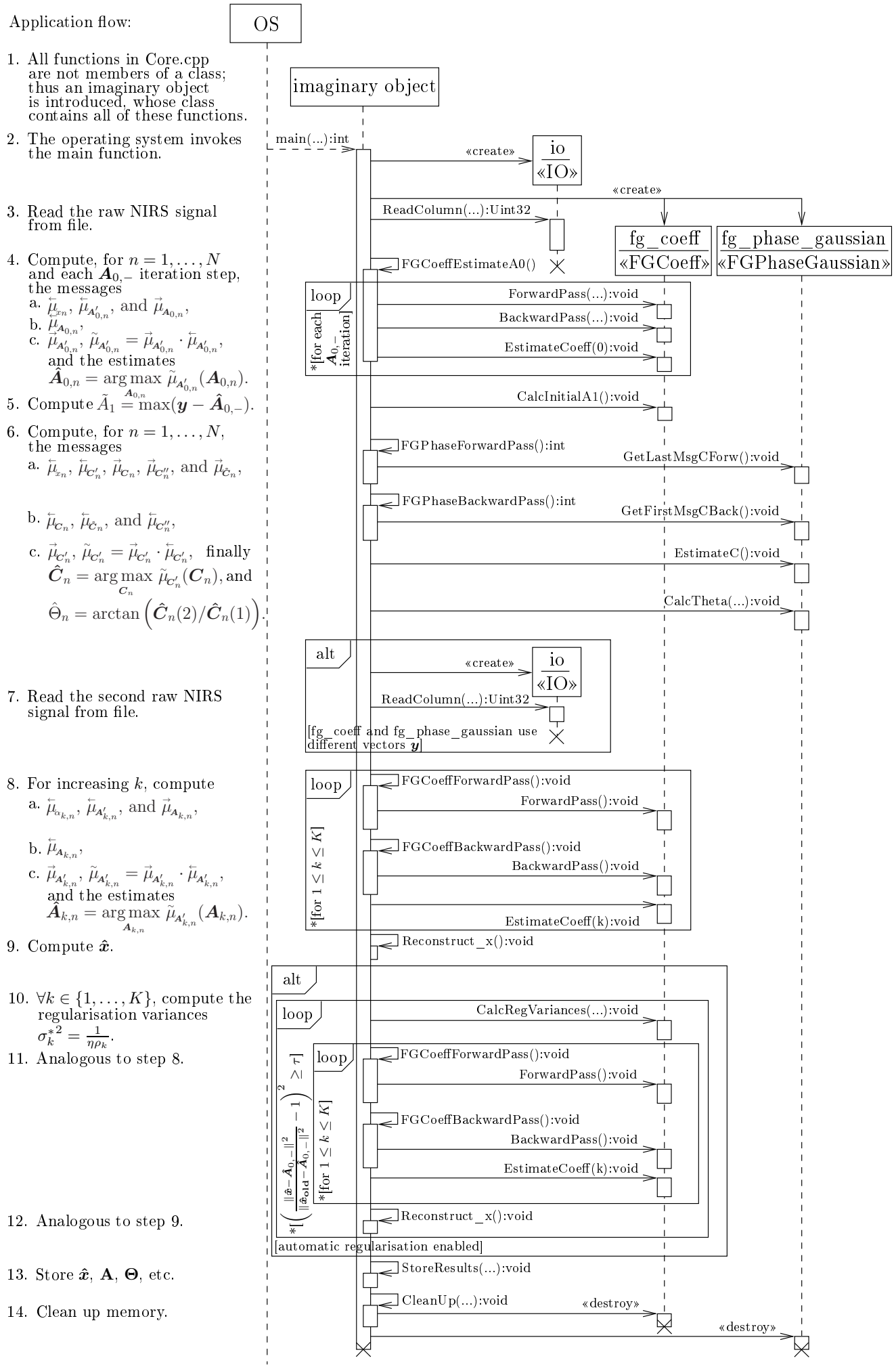

Figure 4.5: The time course of interactions between objects in POETGaussian.

`Gaussian::MakeNeutral()` in Section 3.2.4) and (ii) `ret` referencing a temporary object which is discarded.

- `void FGPhaseBackwardPass()` is equivalent with `::FGPhaseForwardPass(...)`, but calls `FGPhase::GetFirstMsgCBack(...)`.
- `int main(int argc, char* argv[])`, alias `main()`, implements the global schedule described in Section 4.1.2. This method is overviewed in Fig. 4.5. In contrast to POETDiscretePhase, no threads are used. Using them to parallelise computing the message sequences $\vec{\mu}_{\boldsymbol{C}_n} \rightarrow \vec{\mu}_{\boldsymbol{C}''_n} \rightarrow \vec{\mu}_{\mathring{\boldsymbol{C}}_n}$ and $\overleftarrow{\mu}_{\boldsymbol{C}_n} \leftarrow \overleftarrow{\mu}_{\boldsymbol{C}''_n} \leftarrow \overleftarrow{\mu}_{\mathring{\boldsymbol{C}}_n}$ in Fig. 4.2 could further decrease the processing time on processors with multiple cores.
  Note that, 10-12 in Fig. 4.5 is iterated as long as the energy ratio between the reconstructed oscillation from the previous iteration and the one from the current iteration is higher than a threshold τ. The latter is hard-coded in `Core.cpp` (`THRESHOLD_CHANGE_MEAN_ENERGY_PCEST`). If the ratio does not converge, the loop stops after a maximum number of iterations (`::num_of_iterations`).

With command line options `-pci` and `-cci`, different columns (data channels) in the input file can be chosen to represent $\boldsymbol{y}$ when estimating $\boldsymbol{\Theta}$ and $\boldsymbol{A}_{0,-}$, or when estimating $\boldsymbol{A}_{1,-}, \ldots, \boldsymbol{A}_{K,-}$ respectively; $\boldsymbol{y}$ in the former case is read in step 3, and $\boldsymbol{y}$ in the latter case is read in step 7 in Fig. 4.5. This feature is used when estimating the phase in a noisy data channel is not robust. Assuming that the phase is similar in all data channels which is reasonable for the heartbeat component, a data channel with a clear heartbeat component is chosen for estimating the phases; the coefficients are estimated from the noisy data channel and the present phase estimates.

4.3.2 The class `Gaussian`

The following member methods, additional to or different from the ones in Section 3.2.3, should be pointed out.

- `void PlusDiagVar(const Gaussian & msg1, ParameterDataType var)` implements the "+"-node inside the dashed box of Eq. (4.7) in Fig. 4.2. According to table 2 in [21], $\vec{\mathbf{V}}_{\boldsymbol{C}_{n+1}} = \vec{\mathbf{V}}_{\mathring{\boldsymbol{C}}_n} + \vec{\mathbf{V}}$; $\vec{\boldsymbol{m}}_{\boldsymbol{C}_{n+1}} = \vec{\boldsymbol{m}}_{\mathring{\boldsymbol{C}}_n}$, since $\boldsymbol{U}_n$ is zero-mean. Stepwise computing the function composition

$$\vec{\mathbf{W}}_{\mathring{\boldsymbol{C}}_n} \xrightarrow{()^{-1}} \vec{\mathbf{V}}_{\mathring{\boldsymbol{C}}_n} \xrightarrow{+\mathbf{V}} \vec{\mathbf{V}}_{\boldsymbol{C}_{n+1}} \xrightarrow{()^{-1}} \vec{\mathbf{W}}_{\boldsymbol{C}_{n+1}} \tag{4.19}$$

  starting from $\vec{\mathbf{W}}_{\mathring{\boldsymbol{C}}_n} = \left(\begin{smallmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \end{smallmatrix}\right)$ with $\mathbf{V} = \left(\begin{smallmatrix} \sigma_U^2 & 0 \\ 0 & \sigma_U^2 \end{smallmatrix}\right)$ may pose numerical problems, since inverting $\vec{\mathbf{W}}_{\mathring{\boldsymbol{C}}_n}$ depends on $\hat{\boldsymbol{A}}_{\boldsymbol{1}}$: let, for example, $\angle\tilde{A}_1 \rightarrow \frac{\pi}{4}$ and thus $\mathrm{Re}(\tilde{A}_1) \rightarrow \mathrm{Im}(\tilde{A}_1)$. The latter implies that $\det(\overleftarrow{\mathbf{W}}_{\boldsymbol{C}'_n}) \rightarrow 0$; from $\vec{\mu}_{\boldsymbol{C}_1}$ being neutral and hence $\vec{\mathbf{W}}_{\boldsymbol{C}_1} = \mathbf{0}$, it follows that also $\det(\vec{\mathbf{W}}_{\boldsymbol{C}''_1})$ and $\det(\vec{\mathbf{W}}_{\mathring{\boldsymbol{C}}_1})$ converge to 0. By evaluating the composition (4.19) analytically, $\vec{\mu}_{\boldsymbol{C}_{n+1}}$'s weight parameters may be stated directly in terms of $\vec{\mu}_{\mathring{\boldsymbol{C}}_n}$'s weight parame-

ters:

$$\vec{\mathbf{W}}_{C_{n+1}} = \frac{\begin{pmatrix} w_{1,1} + \sigma_u^2 \det(\vec{\mathbf{W}}_{\mathring{C}_n}) & w_{1,2} \\ w_{2,1} & w_{2,2} + \sigma_u^2 \det(\vec{\mathbf{W}}_{\mathring{C}_n}) \end{pmatrix}}{1 + \sigma_u^2 (w_{1,1} + w_{2,2} + \sigma_u^2 \det(\vec{\mathbf{W}}_{\mathring{C}_n}))} \tag{4.20}$$

and

$$\vec{\mathbf{W}}_{C_{n+1}} \vec{\boldsymbol{m}}_{C_{n+1}} = \frac{\begin{pmatrix} 1 + \sigma_u^2 w_{2,2} & -w_{1,2} \\ -w_{2,1} & 1 + \sigma_u^2 w_{1,1} \end{pmatrix} \vec{\mathbf{W}}_{\mathring{C}_n} \vec{\boldsymbol{m}}_{\mathring{C}_n}}{1 + \sigma_u^2 (w_{1,1} + w_{2,2} + \sigma_u^2 \det(\vec{\mathbf{W}}_{\mathring{C}_n}))}. \tag{4.21}$$

$\vec{\mu}_{C_{n+1}}$'s weight parameters (even if $\det(\vec{\mathbf{W}}_{\mathring{C}_n}) = 0$!), are calculated from $\vec{\mu}_{C_n}$'s weight parameters based on Eq. (4.20) and Eq. (4.21). If $\boldsymbol{U}_n$ is viewed as Y in table 2 in [21], then $m_y = 0$ since $\boldsymbol{U}_n$ is zero-mean; then, the rules (II.7) and (II.9) correspond with (II.8) and (II.10). Conclusively, $\overleftarrow{\mu}_{\mathring{C}_n}$'s weight parameters can be derived from $\overleftarrow{\mu}_{C_{n+1}}$'s weight parameters also based on Eq. (4.20) and Eq. (4.21).
The argument `msg1` references the objects holding $\vec{\mu}_{\mathring{C}_n}$ if $\vec{\mu}_{C_{n+1}}$ is computed or the object holding $\overleftarrow{\mu}_{C_{n+1}}$ if $\overleftarrow{\mu}_{\mathring{C}_n}$ is computed respectively. The argument `var` holds σ_u^2.

- `void MatrixMultNode(const Gaussian & msg1, RealParameterMatrix A)` calculates $\overleftarrow{\mu}_X$'s weight parameters according to Eq. (III.5) and Eq. (III.6) in table 3 in [21], if the object referenced by `msg1` is $\overleftarrow{\mu}_Y$. Hermitian conjugate operators H are replaced by transpose operators T, since the multiplication matrix $\mathbf{A}$ (argument `A`) is be real-valued. To compute $\vec{\mu}_Y$, `msg1` must reference the object holding $\vec{\mu}_X$, and `A` must hold the inverted multiplication matrix, since inverting Eq. (III.1) in table 3 in [21] yields

$$\vec{\mathbf{W}}_Y = \left(\mathbf{A}^{-1}\right)^T \vec{\mathbf{W}}_X \mathbf{A}^{-1}, \tag{4.22}$$

and combining Eq. (4.22) and (III.2) in table 3 in [21] yields

$$\vec{\mathbf{W}}_Y \vec{\boldsymbol{m}}_Y = \left(\mathbf{A}^{-1}\right)^T \vec{\mathbf{W}}_X \vec{\boldsymbol{m}}_X. \tag{4.23}$$

- `void RegEqGammaNode(const Gaussian & msg1, const Gaussian & msg2, ParameterDataType gamma, ParameterDataType reg_correction_weight)` implements the rules (2.23) and (2.24) with arguments `msg1` referencing $\vec{\mu}_X$, `msg2` referencing $\overleftarrow{\mu}_Y$, `gamma` holding γ, and `reg_correction_weight` holding σ_k^{*2}.

4.3.3 The class `FGPhaseGaussian`

This class implements the message passing in Fig. 4.2 which is exclusively based on the weight parameters. The following public member methods offer handy services to `main()` (the step numbers refer to the schedule of the message passing algorithm at the end of Section 4.1.2).

- `void FGPhaseGaussian::GetLastMsgCForw(const Gaussian& msg_c_forw_first, Gaussian& ret)` executes step 3 by calling the

private member methods `CalcMsgCCCForw(...)`, `CalcMsgCrotForw(...)` and `CalcMsgCForw(...)`. If $\overleftarrow{\mu}_{C'_n}$ for $n = 1, \ldots, N$ and the matrix $\mathrm{rot}(\hat{\Omega})$ have not been computed yet, i.e. `FGPhaseGaussian::GetFirstMsgCBack(...)` (described below) was not called previously, the private member methods `CalcMsg_x_Back(...)`, `CalcAhat(...)`, `CalcMsgCCBack(...)`, and `CalcRotOmegaMatrix(...)` are called first.
The argument `msg_c_forw_first` references the object holding $\vec{\mu}_{C_1}$; `ret` references the (instantiated!) object where $\vec{\mu}_{C_N}$ is stored. The latter is discarded at the moment, but may be useful in a real time version of POETGaussian.

- `void FGPhaseGaussian::GetFirstMsgCBack(const Gaussian& msg_ccc_back_last, Gaussian ret)` executes step 4 by calling the private member methods `CalcMsgCBack(...)`, `CalcMsgCrotBack(...)` and `CalcMsgCCCBack(...)`. If $\overleftarrow{\mu}_{C'_n}$ for $n = 1, \ldots, N$ and the matrix $\mathrm{rot}(\hat{\Omega})$ have not been computed yet, i.e. `FGPhaseGaussian::GetFirstMsgCForw(...)` (described above) was not called previously, the private member methods `CalcMsg_x_Back(...)`, `CalcAhat(...)`, `CalcMsgCCBack(...)`, and `CalcRotOmegaMatrix(...)` are called first.
  The argument `msg_ccc_back_last` references the object holding $\overleftarrow{\mu}_{C''_N}$; `ret` references the (instantiated!) object where $\overleftarrow{\mu}_{C_1}$ will be stored to. The latter is discarded.
- `void FGPhaseGaussian::EstimateC()` executes step 5 if steps 3 and 4 have been executed; otherwise POETGaussian terminates.
- `void FGPhaseGaussian::CalcTheta(RealParameterVector theta_est_buf) const` calculates $\hat{\Theta}_n$ for $n = 1, \ldots, N$ according to Eq. (4.9).

The following private member methods should be pointed out.

- `void FGPhaseGaussian::CalcMsg_x_Back()` calculates $\overleftarrow{\mu}_{x_n}$ for $n = 1, \ldots, N$ according to step 1. If the value of $y_n - \hat{A}_{0,n}$ is NaN, $\overleftarrow{\mu}_{x_n}$ is set neutral.
- `void FGPhaseGaussian::CalcAhat()` calculates $\hat{\boldsymbol{A}}_1$.
- `void FGPhaseGaussian::CalcMsgCCBack()` calculates $\overleftarrow{\mu}_{C'_n}$ (step 2).
- `void FGPhaseGaussian::CalcRotOmegaMatrix()` calculates the matrix $\mathrm{rot}(\hat{\Omega})$. The value of $\hat{\Omega}$ is passed through the constructor of this class.
- `void FGPhaseGaussian::CalcMsgCCCForw(Uint32 n)` calculates $\vec{\mu}_{C''_n}$ for $n =$`n` (step 3). This method uses `Gaussian::EqNode(...)`.
- `void FGPhaseGaussian::CalcMsgCrotForw(Uint32 n)` calculates $\vec{\mu}_{\hat{C}_n}$ for $n =$`n` (step 3). This method uses `Gaussian::MatrixMultNode(...)`.
- `void FGPhaseGaussian::CalcMsgCForw(Uint32 n)` calculates $\vec{\mu}_{C_n}$ for $n =$`n` (step 3). This method uses `Gaussian::PlusDiagVar(...)`.
- `void FGPhaseGaussian::CalcMsgCrotBack(Uint32 n)` calculates $\overleftarrow{\mu}_{\hat{C}_n}$ for $n =$`n` (step 4). This method uses `Gaussian::PlusDiagVar(...)`.
- `void FGPhaseGaussian::CalcMsgCCCBack(Uint32 n)` calculates $\overleftarrow{\mu}_{C''_n}$ for $n =$`n` (step 4). This method uses `Gaussian::MatrixMultNode(...)`.
- `void FGPhaseGaussian::CalcMsgCBack(Uint32 n)` calculates $\overleftarrow{\mu}_{C_n}$ for $n =$`n` (step 4). This method uses `Gaussian::EqNode(...)`.

- `void FGPhaseGaussian::CalcMsgCCForw()` calculates $\vec{\mu}_{C'_n}$ for $n =$`n` (step 5). This method uses `Gaussian::EqNode(...)`.

4.3.4 Compiling and running POETGaussian

Compiling POETGaussian is similar to compiling POETDiscretePhase. The aspects 3, 8, 9, and 10 in Section 4.3.4 also apply for POETGaussian. The following additional aspects should be pointed out.

1. `./poet_gaussian -h` lists descriptions of all command line options. The relevant ones in this section are: `-cg0` sets γ for $k = 0$, `-cgh` sets γ for $k > 0$, `-oh` sets $\hat{\Omega}$, `-v` sets σ^2 in all factor graphs, `-peuv` sets σ_k^{*2}, `-rs` sets η.
2. If the phase increases $\hat{\Omega}_n = (\hat{\Theta}_{n+1} - \hat{\Theta}_n) \bmod 2\pi$ partly are negative, the diagonal value σ_k^{*2} of $\boldsymbol{U}_n$'s covariance matrix probably is too high. Choosing σ_k^{*2} too low makes $\hat{\Omega}_n$ tense, and the phase estimates $\hat{\Theta}_n$ do not follow the periods in the almost periodic signal.
3. The following command line, mostly working well with the heartbeat component in adults, was intuitively and empirically determined: `./poet_gaussian -v 600 -cg0 0.97 -cgh 0.945 -ifn path/to/foo.csv -pci 18 -cci 18 -dps -peuv 0.00003 -oh 0.0595 -d -rs 0.0005` where the column index 18 in `-pci 18 -cci 18` is replaced by the real one or according to the last paragraph in Section 4.3.1), and `path/to/foo.csv` is replaced by the real file.
4. The heart's interbeat intervals estimated as in Section 3.3.4, but using POETGaussian (command in 3) instead of POETDiscretePhase, have correlations with the corresponding intervals from ECG in the same range as the ones derived using EMD (column "ECG/EMD", Table 3.1). Conclusively, in this specific application, the message passing in Section 4.1.2 yields slightly less precise results than the message passing in Section 3.1.6. A reason could be that the former's estimates are based on the fundamental oscillation only with constant amplitude $\tilde{A}_1$ in Eq. (4.6), whereas the latter's estimates are based on the full coefficient matrix $\hat{\mathbf{A}}$.
5. POETGaussian's computation time (inclusively disk IO) (compiled under Linux with `make opt`) for a 100 s NIRS signal invoked by `./poet_gaussian -v 600 -cg0 0.97 -cgh 0.945 -ifn RawNirsSignals/measurement1.csv -pci 6 -cci 6 -dps -peuv 0.00003 -oh 0.0595 -d -rs 0.0005 -N 10000` is ≈2 s on Intel®'s Core™2Duo @2.66 GHz and ≈7 s on Intel®'s Pentium® 4 @2.4 GHz with 3 iterations in the "Coefficient estimator"-"Regularisation". The number of these iterations considerably impacts the computation time.
6. POETGaussian (command in 5) consumes ≈27 MB of memory.
7. Fig. 4.6 shows results of POETGaussian. The gray curves in all plots are equivalent with the ones in Fig. 3.8. The black curves in A-D were computed using the command in 3. The black curves in E-J were computed similarly, but with `-cg0 0.94 -cgh 0.94 -rs 0.001 -oh 0.125`.

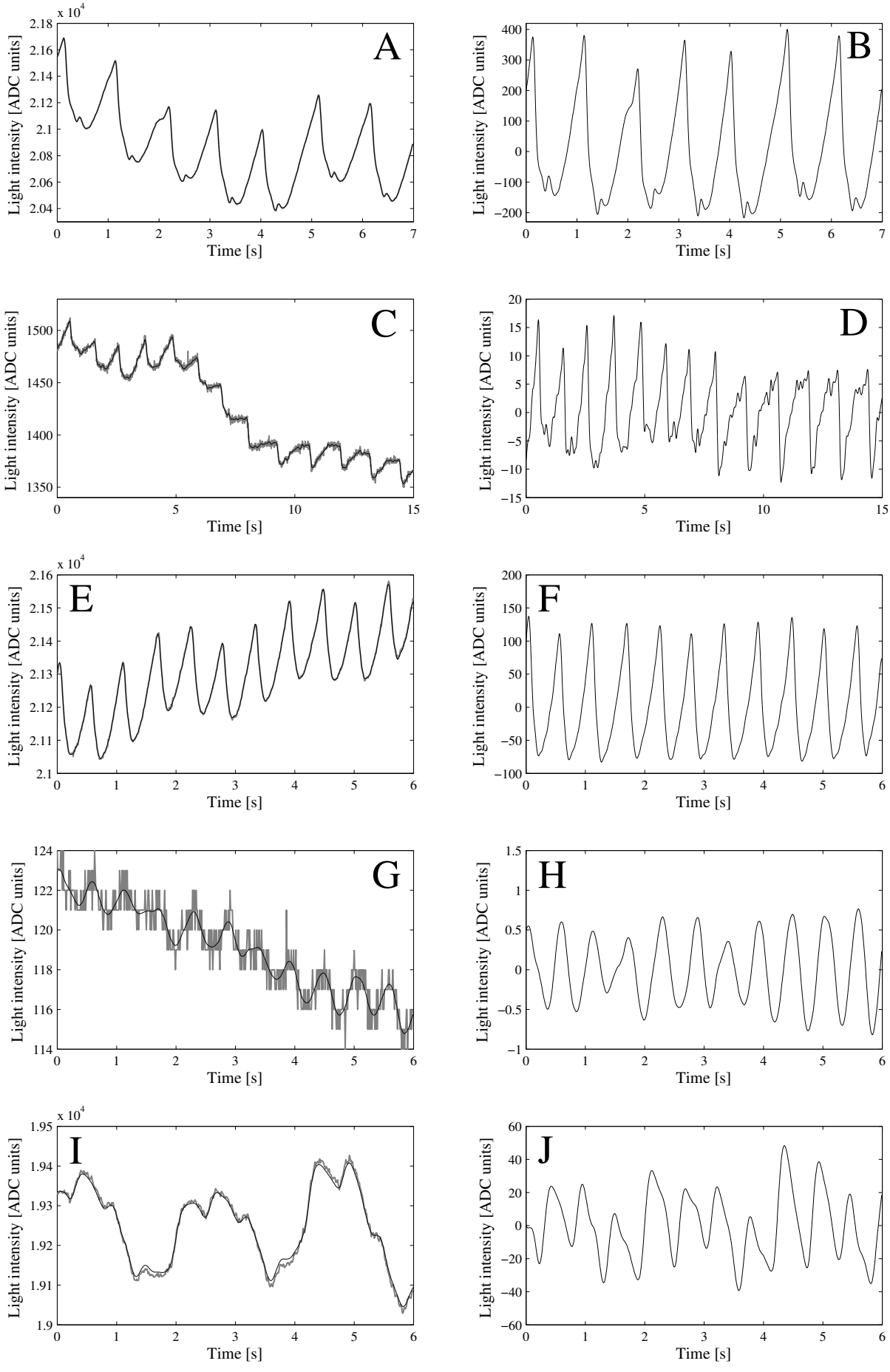

Figure 4.6: The same as Fig. 3.8 except that the reconstructed heartbeat oscillations (black curves) have been computed with POETGaussian instead of POETDiscretePhase.

4.4 Application with optical neuronal signals

Title of the publication
Reproducibility and sensitivity of detecting brain activity by simultaneous electroencephalography and near-infrared imaging

Authors and affilations
Martin Biallas [1,2], Ivo Trajkovic [1,2], Daniel Haensse [1,2], Valentine Marcar [3,4], Martin Wolf [1,2]

[1] Institute for Biomedical Engineering, University of Zurich and ETH Zurich, 8093 Zurich, Switzerland
[2] Biomedical Optics Research Laboratory, Division of Neonatology, Department of Obstetrics and Gynaecology, University Hospital Zurich, 8091 Zurich, Switzerland
[3] Institute of Psychology, Neuropsychology, University of Zurich, Zurich, Switzerland
[4] ZHAW Zurich University of Applied Sciences, Dept. Gesundheit, Physiotherapie, 8401 Winterthur

Status
to be submitted to *Experimental Brain Research*, in the peer-review process.

Abstract
The aims were (i) to determine the sensitivity and reproducibility to detect the hemodynamic responses and optical neuronal signals to brain stimulation by near-infrared imaging (NIRI) and evoked potentials by electroencephalography (EEG), and (ii) to test the effect of novel filters on the signal to noise ratio. This was achieved by simultaneous NIRI and EEG measurements in 15 healthy adults during visual stimulation. Each subject was measured three times on three different days.

The sensitivity of NIRI to detect hemodynamic responses was 57.1 % with novel filtering and 40 % without. The reproducibility in single subjects was low. For the EEG the sensitivity was 86.4 % and the reproducibility 57.1 %. An optical neuronal signal was not detected, although novel filtering considerably reduced noise.

4.4.1 Introduction

Functional near-infrared imaging (NIRI) measures non-invasively changes in oxyhemoglobin ($\Delta[O_2Hb]$) and deoxyhemoglobin ($\Delta[HHb]$) concentrations caused by localised cortical activity of the brain. Although discussed controversially, NIRI may be able to detect the optical neuronal signal [41, 42, 9, 12, 11, 13]. A review on these topics can be found in [43].

The aim of this study was to determine (i) the sensitivity of NIRI to detect visually evoked hemodynamic responses, (ii) the effect of applying a "double detector optode least square approach" (DDOLS) [44] to attenuate superficial physiological signal components, (iii) the reproducibility of these hemodynamic responses in repeated recordings of each subject, (iv) the efficiency of a novel approach "parameter estimation of a model for almost periodic signals" (PEMAPS) to remove the heartbeat in NIRI signals to reduce noise, (v) the sensitivity and reproducibility of optical neuronal signals in NIRI and (vi) the sensitivity and reproducibility of evoked potentials in EEG.

4.4.2 Method: Subjects

Fifteen healthy adult subjects (10 male, 5 female, mean age±SD 29.53±7.89 years) participated in this study. Each subject was measured three times on three different days. Subjects with corrective lenses were asked to wear them during the experiment and were instructed to avoid movement. This study was performed in compliance with the Code of Ethics of the World Medical Association (Declaration of Helsinki) and was approved by the Ethical Committee of the County of Zurich. Written informed consent was obtained from all subjects before inclusion in the study.

4.4.3 Method: Instrumentation

NIRI data was acquired using the multi-channel continuous wave near-infrared imaging device MCPII [4] with the sensor displayed in Fig. 4.7. MCPII was configured to measure 11 source/detector combinations, each of 750 nm, 800 nm and 875 nm resulting in 33 data channels. The term "raw NIRI signal" refers to a single data channel. The sampling rate was 100 Hz per data channel, i.e. every 10 ms, 33 samples were acquired by time-multiplexing [4, 2].

The sensor was placed on the head such that the electrode's position O_1 (according to the international 10/20 system [45]) was located as shown in Fig. 4.8.

To reduce light attenuation by hair, a stencil of the sensor with holes at all source/detector positions was placed at the appropriate position and fixated tightly with stripes of Velcro. Hair under the stencil's holes was tugged aside by cotton buds. Finally, the sensor was placed over the stencil and attached to the head by bandages.

EEG data were recorded with MitSar201® using the WinEEG software. Ag/AgCl ring electrodes in conjunction with abrasive paste were used to improve skin conductivity.

Before and after recording, electrode impedances were assured to be below 15 kΩ. Electrodes were positioned according to [45] at O_2, F_3, ground at F_Z and the reference at the earlobe. To minimise electrical interference between the electrodes and the NIRI sensor, MCPII's and MitSar®'s amplifiers were put on opposite sites of the subject, such that the paths of the electrode leads and the sensor's cable were in opposite directions.

The subject's visual cortex was stimulated by a TFT screen (250 cd/m^2, full on/full off contrast ratio 400:1). A separate computer was attached to this

screen and generated synchronising signals for MCPII and MitSar®.

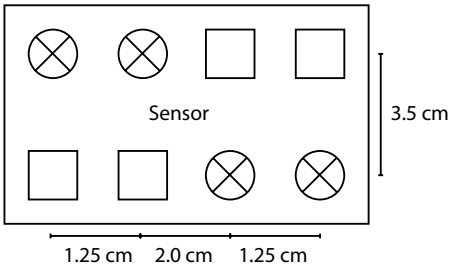

Figure 4.7: Geometry of the used sensor. Light sources/detectors are circles/squares.

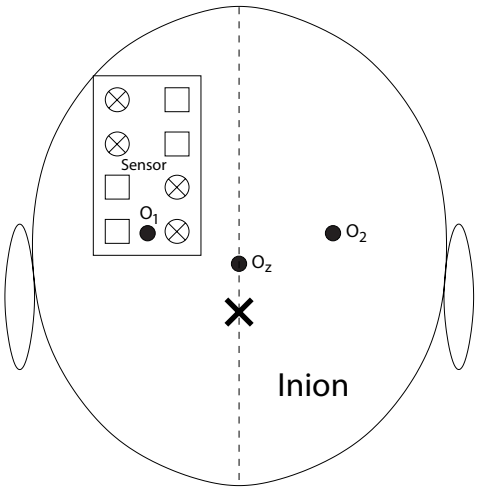

Figure 4.8: Sensor's positioning on the back of the subject's head. Electrode O_1 was not placed. EEG analysis was performed on O_2 only.

4.4.4 Method: Protocol

During the experiment, subjects were lying on a table with an u-shaped headrest which permitted unobstructed view to the floor, where a TFT-monitor was installed subtending a visual angle of $18.7° \times 24.2°$ for visual stimulation. Stimulation was by black and white dartboard pattern reversals and a black screen during rest intervals programmed in Presentation®. Stimulation intervals and rest intervals were 20 s long. Before the start of an experiment, the frequency of the pattern reversal was adjusted to 1.5 or 2.5 times the heart rate to reduce noise. The frequency of the pattern reversal was varied randomly by ± 0.5 Hz. The experiment took 20 min inclusively 2 min baseline recording before the first stimulation interval and 1 min baseline after last stimulation interval. During the whole time, the room was kept dark and quiet.

4.4.5 Method: Analysis of hemodynamic response of NIRI

A raw NIRI signal is a time series in which each element is proportional to the measured light intensity. A measurement consists of (i) raw NIRI signals, one for each source/detector/wavelength combination, (ii) signals which represent ambient light and (iii) event markers. From the 3 raw NIRI signals of the 3 wavelengths from a specific source/detector combination, the sensor's source/detector geometry and the modified Beer-Lambert law [46, 4], 2 new

signals can be derived; i.e. temporal concentration changes in oxyhemoglobin ($\Delta[O_2Hb]$) and deoxyhemoglobin ($\Delta[HHb]$).

Our setup yielded 11 $\Delta[O_2Hb]$ or $\Delta[HHb]$ signals. Absorption coefficients were taken from the UCL's website[1], and the differential path length factor (DPF) was set to 8.24, 7.84 and 7.29 at 750 nm, 800 nm and 875 nm [47].

A customised algorithm implemented in Matlab® evaluated the measurements. This algorithm incorporates deriving $\Delta[O_2Hb]$ and $\Delta[HHb]$ and filtering by DDOLS. The shortest source/detector distance was 20 mm. All channels were high-pass filtered (f_c=0.025 Hz). Exceptional fluctuations in channels, due to movement artifacts, were identified for $\Delta[O_2Hb]$ and $\Delta[HHb]$ signals separately by the following procedure: When a sample in the high-pass filtered version of the signal (5 pole IIR Butterworth, cutoff frequency 0.5 Hz) exceeded 2 µmol/l, neighbouring samples up to 1 s before and 3 s after this sample and the sample itself were excluded from further evaluation.

For each channel, the last 10 s before the stimulation and the period of 10 s to 20 s after the beginning of the stimulation were statistically compared by a paired Wilcoxon test. A hemodynamic response was detected, if the values differed significantly with $p<0.05$.

4.4.6 Method: EEG data analysis

Only the signal at electrode O_Z was considered. This signal was filtered (band-pass with cutoff frequencies 0.32 Hz and 70 Hz, then notch between 45 Hz and 55 Hz) during recording. When a sample in the signal exceeded 200 µV, neighbouring samples and the sample itself were excluded from the evaluation.

In the next step, the signal was detrended by band pass filtering (5-pole IIR Butterworth).

For stimulation or sham events, the last 50 ms before and the interval from 125 ms to 175 ms after the stimulation event were compared by a paired Wilcoxon test. A visual activation was detected when the values from the stimulation events differed significantly, and the ones from sham events did not ($p<0.05$). Testing both types of events prevents the detection of false-positive activations caused by electromagnetic interference.

4.4.7 Method: Analysis of optical neuronal responses

The analysis consists of the following steps.

1. Estimating the heartbeat component in each data channel by PEMAPS (Section 4.4.8). For all subjects, PEMAPS was set up with $K \triangleq 15$ in (4.24), message damping factors for harmonic indices $k = 0$: $\gamma \triangleq 0.97$, $k > 0$: $\gamma \triangleq 0.945$ (see [40], section "The new coefficient estimator"); furthermore, $\hat{\Omega} \triangleq 0.0595$ rad. (see paragraph after Eq. (4.30)), regularisation strength $\eta \triangleq 0.0005$ and $\sigma_k^{*2} = 0.00003$ (see [40], section "The new coefficient estimator"), and the variance of the "$\mathcal{N}$"-node in all factor graphs (see [40]) $\sigma^2 \triangleq 600$. These values were determined empirically.

[1] `http://www.medphys.ucl.ac.uk/research/borl/research/NIR_topics/spectra/spectra.htm`

2. Calculating the corresponding optical density signal from each data channel: $\boldsymbol{o} \triangleq \frac{\log(\boldsymbol{y}-\hat{\boldsymbol{x}}+\hat{\boldsymbol{A}}_{0,-})}{\mathrm{DPF}(\lambda)\cdot d}$ with geometric source/detector distance d (in cm), wavelength λ, $\boldsymbol{y}$ and $\hat{\boldsymbol{x}}$ as defined in Section 4.4.8, and a slow trend $\hat{\boldsymbol{A}}_{0,-}$. The latter was computed separately with $\gamma \triangleq 0.999$ to minimise the influence of the heartbeat's fundamental frequency.
3. Bandpass filtering of $\boldsymbol{o}$ with cutoff frequencies < 5 Hz and 40 Hz. The low cutoff frequency was manually calculated for each subject to ensure that it was lower than the varying frequency of the pattern reversals in the visual stimulation (see Section 4.4.4). The higher cutoff frequency was chosen empirically to attenuate irrelevant frequencies.
4. Subtracting the mean value of each segment (stimulus and sham) in $\boldsymbol{o}$.
5. Applying a moving variance window to identify segments which exceeded the threshold of two times the variance of all segments. In each data channel, this procedure identified and rejected stimulation (sham) events with outliers caused by movement artefacts.
6. Computing two average segments for each data channel; one for all accepted stimulation events and one for all accepted sham events. These segments are based on 200 ms long signals.

The shape of optical neuronal responses is unknown. Thus, each sample in the 200 ms long average segment was checked if it significantly differs from 0 (t-test, $p<0.05$). This was done for all source/detector/wavelength combinations and subjects.

4.4.8 Method: Filtering the heartbeat by PEMAPS

The heartbeat component in raw NIRI signals is a tremendous source of noise in the analysis of the optical neuronal signal, because changes in $\boldsymbol{o}$ due to the heart beat are considerably larger than the expected size of the optical neuronal signal [48]. The aim was to estimate the pure heartbeat component and to subtract it from the raw NIRI signal.

Since there are sharp peaks in the heartbeat, a simple low-pass filter will not work [30]; thus, we used the method for modelling and adaptive filtering of oscillatory components called Parameter Estimation of a Model for Almost Periodic Signals (PEMAPS) [49] which is summarised here.

The heartbeat component is not strictly periodic; it can be characterised as "almost periodic". In an almost periodic signal, the period length and the signal shape drift over time [30]. Such a signal can be described by a "Fourier series" with time-variant fundamental frequency (related to the varying period length) and time-variant coefficients (related to the varying signal shape). Under this assumption, the sampled version of the real-valued heartbeat component $x_1, x_2, \ldots$ is given as

$$x_n = \mathrm{Re}\left(\sum_{k=1}^{K} A_{k,n} e^{jk\Theta_n}\right) \tag{4.24}$$

with coefficients $A_{0,n} \in \mathbb{R}$, $A_{1,n}, \ldots A_{K,n} \in \mathbb{C}$, phase $\Theta_n \in [0, 2\pi]$, finite number

of frequencies K and

$$A_{k,n+1} \approx A_{k,n}, \tag{4.25}$$
$$\Theta_{n+1} = (\Theta_n + \Omega_n) \bmod 2\pi, \tag{4.26}$$
$$\Omega_{n+1} \approx \Omega_n. \tag{4.27}$$

Eq. (4.27) expresses the varying heart rate; (4.25) expresses the varying beat shape. Let the raw NIRI signal vector $\boldsymbol{y} = (y_1, \ldots, y_N)$ be a noisy, trended version of $\boldsymbol{x} = (x_1, \ldots, x_N)$ where N is the signal length. Specifically,

$$\boldsymbol{y} = \boldsymbol{A}_{0,-} + \boldsymbol{x} + \boldsymbol{Z} \tag{4.28}$$

where $\boldsymbol{Z} = (Z_1, \ldots, Z_N)$ is discrete time white Gaussian noise, and $\boldsymbol{A}_{0,-} = (A_{0,1}, \ldots, A_{0,N})$ models changes slower than the heartbeat and thus is omitted in (4.24). We will use the vectors $\boldsymbol{A}_{k,-} = (A_{k,1}, \ldots, A_{k,N})$ for $k = 0, \ldots, K$ and decorate estimates with a hat (e.g. $\hat{C}$ is an estimate of C).

Given $\boldsymbol{y}$, the objective is to estimate the phases $\boldsymbol{\Theta} \triangleq (\Theta_1, \ldots, \Theta_N)$, K and the coefficient vectors $\boldsymbol{A}_{1,-}, \ldots, \boldsymbol{A}_{K,-}$ such that

$$\sum_{n=1}^{N} (y_n - \hat{x}_n - \hat{A}_{0,n})^2.$$

is minimal, where $\hat{x}_n$ is the reconstructed signal by applying the estimates in (4.24).

The estimation algorithm consists of several building blocks (see Fig. 4.9).

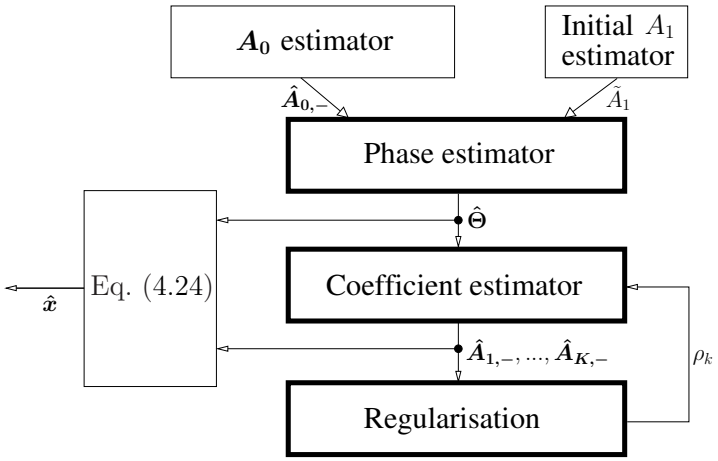

Figure 4.9: This diagram visualises the principle of PEMAPS and its building blocks.

Initially, the "A_0 estimator" estimates the slow trend $\boldsymbol{A}_{0,-}$ by a one-time procedure similar to low pass filtering and based on $\boldsymbol{y}$ only.

In the heartbeat component, most of the energy, apart from the noise, lies in the fundamental frequency coefficient $\boldsymbol{A}_{1,-}$. Thus, a first rough estimate of the heartbeat component is a complex sinusoid with constant complex magnitude. The "Initial A1 estimator" block makes an estimate $\tilde{A}_1$ of this magnitude such that the sinusoid has approximately the same energy as $\boldsymbol{y} - \hat{\boldsymbol{A}}_{0,-}$.

The "Phase estimator" calculates the final estimate $\hat{\boldsymbol{\Theta}}$ of $\boldsymbol{\Theta}$ based on estimates $\hat{\boldsymbol{A}}_{0,-}$ and $\tilde{A}_1$ and (4.24) with $K = 1$ parameterised as

$$\begin{aligned} x_n &= \mathrm{Re}\left(\tilde{A}_1 \cdot e^{j\Theta_n}\right) \\ &= \mathrm{Re}(\tilde{A}_1)\cos(\Theta_n) - \mathrm{Im}(\tilde{A}_1)\sin(\Theta_n) \\ &= \hat{\boldsymbol{A}}_1 \cdot \boldsymbol{C}_n \end{aligned} \tag{4.29}$$

with constant vector $\hat{\boldsymbol{A}}_1 = \Big(\mathrm{Re}(\tilde{A}_1), -\mathrm{Im}(\tilde{A}_1)\Big)$, state vector $\boldsymbol{C}_n = (\cos(\Theta_n), \sin(\Theta_n))^T$ and state transition

$$\boldsymbol{C}_n = \mathrm{rot}(\hat{\Omega}) \cdot \boldsymbol{C}_{n-1} + \boldsymbol{U}_n \tag{4.30}$$

where

$$\mathrm{rot}(\Omega) = \begin{pmatrix} \cos(\hat{\Omega}) & -\sin(\hat{\Omega}) \\ \sin(\hat{\Omega}) & \cos(\hat{\Omega}) \end{pmatrix}$$

is a rotation matrix, and $\hat{\Omega}$ is an a-priori estimate of Ω_n in (4.27). $\hat{\Omega}$ is derived by using the formula in [30], section 3.3, paragraph 4 and assuming a typical heart rate depending on the subject, e.g. H = 80 bpm for adults. Since $\hat{\Omega}$ is fixed, despite the fact that the heart rate varies considerably depending on various factors, uncertainty, i.e. two-dimensional zero-mean white Gaussian noise $\boldsymbol{U}_n$, is added to the rotated state in (4.30). This addition of noise defines (4.27).

The estimate $\hat{\boldsymbol{C}}_n$ of $\boldsymbol{C}_n$ is made as

$$\hat{\boldsymbol{C}}_n = \underset{\boldsymbol{C}_n}{\arg\max}\ f(\boldsymbol{C}_n \mid \boldsymbol{A}_{0,-}, \tilde{A}_1, \boldsymbol{y}). \tag{4.31}$$

The function f in (4.31) (i) comprises (4.28), (4.29) and (4.30) and (ii) is derived with the message passing algorithm described in [49], section 2.

Each estimate $\hat{\Theta}_n$ in $\hat{\boldsymbol{\Theta}}$ is made as

$$\hat{\Theta}_n = \arctan \frac{\hat{\boldsymbol{C}}_n(2)}{\hat{\boldsymbol{C}}_n(1)} \tag{4.32}$$

with $\hat{\boldsymbol{C}}_n(i)$ denoting the i-th entry of the vector $\hat{\boldsymbol{C}}_n$.

The "Coefficient estimator" calculates the full set of coefficient estimates $\hat{\boldsymbol{A}}_{1,-}, \ldots, \hat{\boldsymbol{A}}_{K,-}$. Each estimate $\hat{A}_{k,n}$ of $A_{k,n}$ is calculated based on the estimates $\hat{\boldsymbol{A}}_{k-1,-}, \ldots, \hat{\boldsymbol{A}}_{0,-}$, $\hat{\boldsymbol{\Theta}}$ and $\boldsymbol{y}$ as

$$\hat{\boldsymbol{A}}_{k,n} = \underset{A_{k,n}\in\mathbb{C}}{\arg\max}\ f(A_{k,n} \mid \hat{\boldsymbol{A}}_{k-1,-}, \ldots, \hat{\boldsymbol{A}}_{0,-}, \hat{\boldsymbol{\Theta}}, \boldsymbol{y}) \tag{4.33}$$

for increasing k. The function f in (4.33) (i) comprises (4.24), (4.25) and the assumption of white Gaussian noise in (4.28) and (ii) is derived with the message passing algorithm described in [40], section "The new coefficient estimator".

The "Regularisation"-block is used to iteratively derive the number of harmonics K in (4.24) as described in [49], at the end of section 2.

The "Eq. (4.24)" block reconstructs the heartbeat component by applying the estimates $\hat{\boldsymbol{A}}_{1,-}, \ldots, \hat{\boldsymbol{A}}_{K,-}$ and $\hat{\boldsymbol{\Theta}}$ in (4.24).

4.4.9 Results: hemodynamic response

In 13.6 % there was no activation in the EEG signal, and we concluded that in these subjects, the stimulation was not successful. Using DDOLS a significant hemodynamic response reflecting brain activation was found in 57.1 % of the measurements (16 HR in 28 recordings). Without DDOLS it was lower, i.e. 36.8 % (14 HR in 38 recordings). In 40.0 % (based on 5 subjects) a significant hemodynamic response was found in three repeated measurements with DDOLS and in 12.5 % out of 8 subject without DDOLS. An activation was found at least twice with DDOLS in 30.0 % (based on 10 subjects), and in 26.7 % (based on 15 subjects) without DDOLS. At least one single occurrence of a significant hemodynamic response was found in 78.6 % out of 14 subjects with DDOLS, and 60.0 % out of 15 subjects without DDOLS. Table 4.1 displays the findings for all measurement separately. Fig. 4.10 shows an example of a significant hemodynamic response.

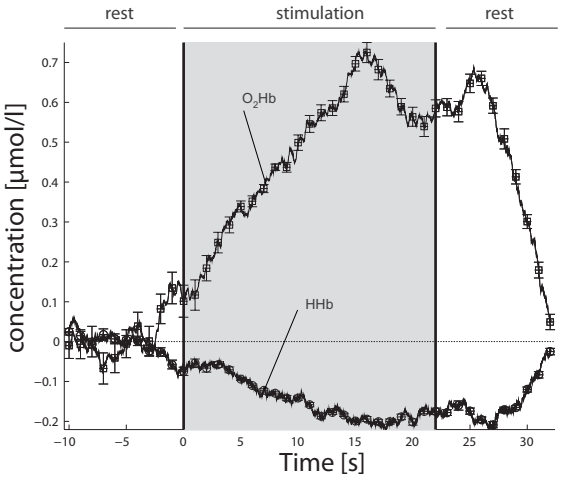

Figure 4.10: A hemodynamic response which shows an increase in $\Delta[O_2Hb]$ and decrease in Δ[HHb]. It was derived by calculating, sample-wise, the median of all accepted stimulation intervals in an $\Delta[O_2Hb]$ and Δ[HHb] signal. Both curves were smoothed by a moving average filter (window length 5 s). The gray area indicates the stimulation period.

4.4.10 Results: Visual evoked potential (VEP)

An exemplary VEP signal is depicted in Fig. 4.11. Assuming that stimulation was successful in all valid measurements yields a sensitivity of 86.4 %. Reproducibility was found in three repeated measurements in 57.1 % of all 14 subjects.[2] In all 15 subjects, a VEP was detected twice. No subject showed a VEP only once or not at all in the repeated measurements.

4.4.11 Results: Optical neuronal signal

The histograms in Fig. 4.13 show the number of significances (y-axis) normalised by the total number of data samples at each point in time (x-axis) after the stimulation. The data are shown for the analysis without (Fig. 4.13a,b) and with PEMAPS (Fig. 4.13c,d) and for data reflecting real stimulations (Fig. 4.13a,c) and sham (Fig. 4.13b,d). For a significance level of $p<0.05$, we would expect a proportion of 0.05 of the signals to be significant just by

[2]One measurement could not be evaluated. Therefore, this subject is not included in the total number of subjects.

Subject index	Gender	Age	Recording index	# days since rec. A	p value HR w/o DDOLS	HR Δmean $[\mu\frac{mol}{l}]$ w/o DDOLS	p value HR w DDOLS	HR Δmean $[\mu\frac{mol}{l}]$ w DDOLS	VEP Δmean [μV]
1	F	21	A	0	-	-	0.0316	0.0192±0.0082	5.8±0.4
			B	1	-	-	0.0207	0.0444±0.0169	7.1±0.2
			C	2	-	-	0.0495	-0.0242±0.0116	2.5±0.4
2	F	25	A	0	0.0313	0.7178±0.0648	n.a.	n.a.	-7.3±0.3
			B	9	-	-	n.a.	n.a.	-8.5±0.3
			C	98	0.0060	0.0838±0.0265	0.0073	0.0723±0.0268	2.8±0.3
3	F	25	A	0	-	-	-	-	4.3±0.3
			B	3	0.0300	-0.0274±0.0161	0.0011	0.0203±0.0056	3.8±0.3
			C	4	-	-	-	-	1.9±0.3
4	F	23	A	0	0.0125	-0.0313±0.0119	$4.4493{\cdot}10^{-5}$	0.0390±0.0069	3.5±0.3
			B	3	0.0068	-0.0280±0.0092	0.0010	-0.0333±0.0088	6.0±0.3
			C	5	0.0333	-0.0216±0.0107	0.0270	0.0078±0.0033	6.0±0.4
5	F	38	A	0	0.0059	0.0999±0.0293	n.a.	n.a.	6.7±0.4
			B	7	0.0020	0.6141±0.0758	n.a.	n.a.	6.0±0.5
			C	60	-	-	0.0449	-0.0430±0.0216	7.0±0.5
6	M	33	A	0	-	-	0.0230	0.0834±0.0339	5.5±0.2
			B	1	-	-	-	-	4.6±0.2
			C	2	-	-	-	-	6.1±0.9
7	M	28	A	0	-	-	-	-	-
			B	1	0.0057	-0.1064±0.0334	n.a.	n.a.	8.4±0.3
			C	7	-	-	-	-	5.6±0.4
8	M	25	A	0	-	-	-	-	2.8±0.3
			B	1	-	-	-	-	-
			C	2	-	-	0.0117	0.0346±0.0121	2.9±0.4
9	M	25	A	0	-	-	-	-	7.6±0.3
			B	4	0.0148	-0.0692±0.0281	0.0077	-0.0308±0.0106	10.8±0.3
			C	6	-	-	-	-	8.0±0.5
10	M	27	A	0	-	-	-	-	3.5±0.3
			B	7	-	-	-	-	2.4±0.4
			C	8	-	-	-	-	-
11	M	28	A	0	-	-	n.a.	n.a.	-
			B	80	-	-	-	-	3.4±0.3
			C	83	0.0117	0.1980±0.0658	-	-	10.2±0.3
12	M	28	A	0	0.0020	-0.0715±0.0140	n.a.	n.a.	8.2±0.2
			B	1	0.0087	0.0873±0.0397	n.a.	n.a.	n.a.
			C	8	-	-	0.0036	0.0751±0.0285	18.3±0.3
13	M	29	A	0	0.0093	-0.0563±0.0189	0.0028	0.0704±0.0202	-
			B	1	-	-	0.0256	0.0399±0.0171	5.5±0.1
			C	4	-	-	0.0449	0.0472±0.0237	5.0±0.2
14	M	38	A	0	-	-	-	-	9.0±0.2
			B	1	-	-	n.a.	n.a.	7.0±0.2
			C	3	-	-	0.0185	0.0174±0.0070	6.5±0.2
15	M	50	A	0	0.0385	-0.1750±0.0725	n.a.	n.a.	6.2±0.3
			B	16	-	-	n.a.	n.a.	-
			C	43	0.0461	-0.1208±0.0543	n.a.	n.a.	5.9±0.4

Table 4.1: HR: hemodynamic response

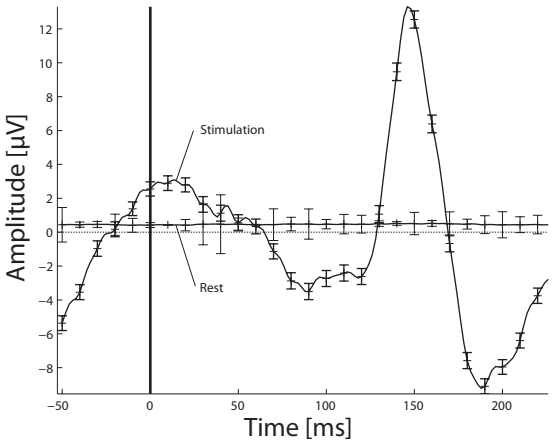

Figure 4.11: Depicted are two signals; one is the block average during stimulation, the other during sham. The VEP is clearly visible as a peak at 150 ms. The pattern reversal happened at 0 ms (vertical line).

chance. Would the optical neuronal response behave like the VEP and feature a peak at 150 ms, the number of detected significances should increase at 150 ms for example. This is however not the case.
Without PEMAPS many significances were found similarly for stimulation and rest. Thus, PEMAPS was able to reduce the number of false positives considerably. Comparing Fig. 4.13c and 4.13d shows that the pattern of significant samples is similar during stimulation and rest meaning that the optical neuronal signal was not detected.

The magnitude of changes in the optical density of a data channel needed to be induced by an optical neuronal signal in order to be detected can be derived from Fig. 4.12. The latter displays the noise level of the measurements with and without PEMAPS. PEMAPS particularly decreased the SEMs of the data channels with already small SEMs ($< 10^{-7}$).

Histograms as in Fig. 4.12 and 4.13 were plotted for each distance/wavelength combination separately to test whether any of the specific wavelengths or distances may be particularly sensitive to detect an optical neuronal signal. In general, these histograms showed a higher influence of the heartbeat on the short distance and long wavelength (20 mm, 875 nm). However, the histograms looked similar for stimulation and rest, and thus none of the specific wavelengths or distances were sensitive enough to detect the optical neuronal signal.

4.4.12 Discussion: Hemodynamic response

The principal idea behind DDOLS is to attenuate superficial and physiological signal components in NIRI data [44]. This is achieved by removing changes in a reference channel with short interoptode distance which mostly detects superficial tissue from channels with longer interoptode distance, which are sensitive to deeper tissue. The shortest interoptode distance was saturated (too much light) in several measurements, such that DDOLS could not be applied. [3] Our shortest interoptode distances of 20 mm was somewhat larger than the

[3]When DDOLS could not be applied, the columns are marked by "n.a." in table **??**.

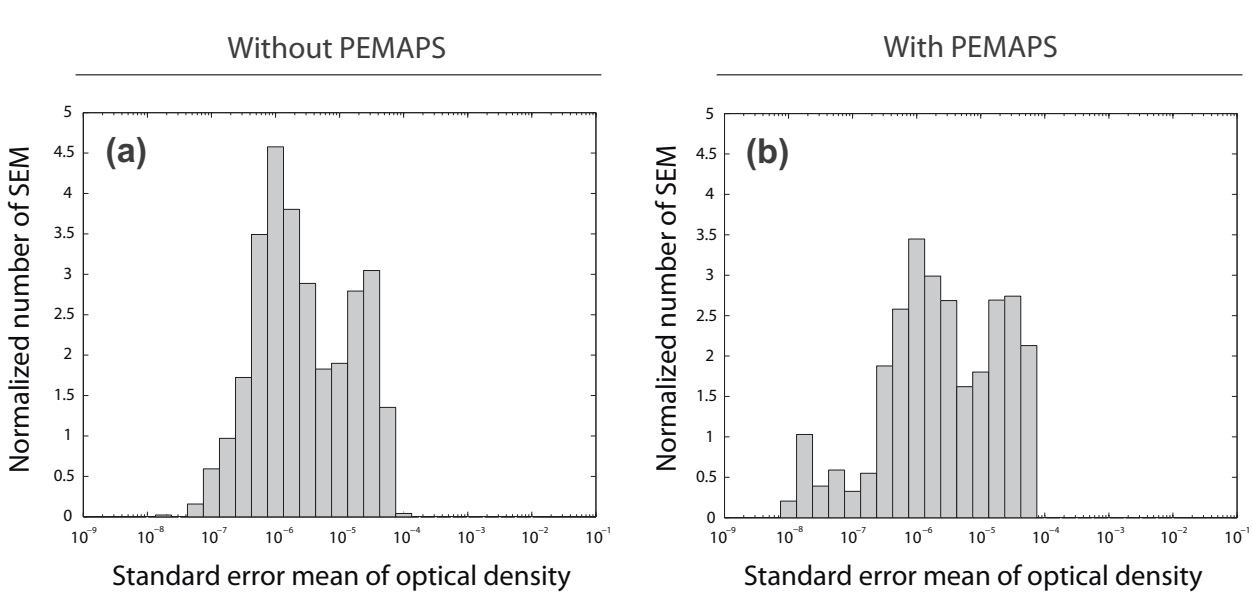

Figure 4.12: Depicted is the distribution of standard errors of the mean (SEM) of all measured data channels. The histogram displays the number of occurrences of a range of SEMs normalised to the number of data channels. PEMAPS increased the number of occurrences of lower SEMs.

1 mm to 13 mm used in [44]. Despite these limitations, DDOLS increased the sensitivity from 36.8 % to 57.1 %. When longer source/detector distances are used for DDOLS' reference channel as in our case, it is advisable to check the reference channels for activation, and account for it in analysis. If activation is present, DDOLS should not be used, because it will remove also the activation signal from other channels[4]. The sensitivity could be increased further with an optode optimised for handling both, the established and very short interoptode distances.

Possible reasons for this may be imprecision in locating the position O_1 [50], repositioning inaccuracy for the sensor between measurements ($\sim \pm 5$ mm), and the area covered by the light sensor. As outlined in [51], location of significantly activated brain areas were only in 55 % identical within a subject over 2 measurements. This is confirmed by [52].

4.4.13 Discussion: Visual evoked potential

A VEP was found in 86.4 % of 44 measurements. One dataset could not be processed due to instrumental artifacts.

Surprisingly, a VEP could not be detected more frequently as suggested by the literature [53, 54]. This may be due to the higher electrode/skin impedances accepted in our study in comparison to other studies [55–57].

This favores the pick up of electromagnetic interference and may obscure small VEPs. A further difference to literature is utilizing only a single channel as reference. Another reason may be that the stimulation was not successful. In our opinion, this is the most probable reason.

4.4.14 Discussion: Optical neuronal signal

Despite low noise levels (Fig. 4.12b), no optical neuronal signal was detected (Fig. 4.13). The idea behind the histograms in Figure 4.13c,d is to compare ran-

[4]This is the case in the following 3 recordings: subject 7, recording B, and in subject 12, recordings A and B.

dom significances to possible event related significances. The random signals were generated by adding stimulation markers during the rest periods when no stimulation occurred. These random signals were analysed in the same way as signals during stimulation, i.e. they include the same instrumental and methodical artifacts. At significance level of $p<0.05$, it is expected that 5 % of the data points are significant just by chance. In Fig. 4.13a and 4.13b higher incidences of significant data points are shown. These higher numbers are the same for real (Fig. 4.13a) and sham (Fig. 4.13b) stimulation clearly indicating that the higher incidence is due to methodological problems and that these incidences are false positive. This illustrates the importance of an effective surpression of the heartbeat component. Without such a filter, the data in the block averaged segments are dominated by artifacts (Fig 4.13a,b). This also demonstrates the importance of proper controls to interpret the results correctly, because otherwise, if only Fig. 4.13a was observed, one may erroneously conclude that optical neuronal signals were present.

In Figures 4.13c and 4.13d the data after removing the heartbeat component by PEMAPS, the incidence of significances reaches a level closer to 5 % which corresponds to the level expected to be significant by chance.

The peaks at about 50 ms after stimulation-onset are again similar for real stimulation and sham and thus do not represent a specific stimulation response. They are probably due to an instrumental artifact.

As already stated, the figures demonstrate the need to compare stimulation and rest intervals to distinguish between real signals and artifacts.

Figure 4.12 displays how noise, i.e. the SEM of optical densities, spreads over several magnitudes. Obviously, there are more and less noisy data channels which depends on the light intensity at the detector, i.e. the more light, the higher the SNR, and the level of physiological noise. For shorter distances, the SNR is higher, since more light is detected. The percentile P_{20} of the histogram in Fig. 4.12b is at $\mathrm{SEM}(\boldsymbol{o}) = 4.8 \cdot 10^{-7}$. This means that in 20 % of the data channels, an optical neuronal signal with an amplitude of $1.96 \cdot \mathrm{SEM}(\boldsymbol{o}) \approx 9.41 \cdot 10^{-5}\%$ would have been detected with significance. Optical neuronal signals with light intensity changes of 0.05 %, as seen in the literature [43], would have been detected.

4.4.15 Conclusion

Applying DDOLS increases the sensitivity of NIRI to detect hemodynamic responses from 36.8 % to 57.1 %. The reproducibility of the hemodynamic response in a single subject was low (40 %). EEG had a high sensitivity of 86.4 % and a reproducibility of 57.1 %. Raw data with low SEM is further enhanced by PEMAPS. Without PEMAPS, artifactual signals are obtained. We detected no optical neuronal signal despite 20 % of the signals having extremely low noise ($4.8 \cdot 10^{-5}\%$). The results underline the importance of a proper control, i.e. analysing rest intervals.

4.4.16 Acknowledgements

The authors gratefully acknowledge the funding of the Swiss National Science Foundation. Many thanks to Felix Scholkmann for help with the figures.

End of publication

4.5 Detectability of optical neuronal signals with POETGaussian

The following question was investigated (independently of the study above): How small can an optical neuronal signal be for just being detected/significant ($p < 0.05$). Three raw NIRS signals with different noise levels and heartbeat components were chosen for this test. A stimulation pattern with 2300 (realistic value) events/segments was generated according to Section 1.3. In each of the three signals, the optical neuronal signal was simulated by increasing the 10-th sample value in each segment by a constant magnitude ε. After applying POETGaussian (command in 3, Section 4.3.4) on each of these three (modified) raw NIRS signals, all segments in each resulting residual signal were averaged according to Section 1.3.

$\bar{\boldsymbol{y}}$	ε	$\sigma_{\boldsymbol{y}}$	$\sigma_{\hat{\boldsymbol{z}}}$	σ_h
1479.2	0.9	15.26	1.88	1.89
16687.23	0.5	611.14	2.66	119.97
21298.89	0.5	892.07	3.21	184.38

Table 4.2: These are the results. $\bar{\boldsymbol{y}} \triangleq$ mean value (in ADC units) of the raw NIRS signal $\boldsymbol{y}$; $\varepsilon \triangleq$ minimal magnitude (in ADC units) of the optical neuronal signal; $\sigma_{\boldsymbol{y}} \triangleq$ the sample standard deviation of $\boldsymbol{y}$; $\sigma_{\hat{\boldsymbol{z}}} \triangleq$ sample standard deviation of the residual signal $\hat{\boldsymbol{z}} = \boldsymbol{y} - \hat{\boldsymbol{x}}$; $\sigma_h \triangleq$ sample standard deviation of the reconstructed heartbeat $\hat{\boldsymbol{x}} - \hat{\boldsymbol{A}}_{0,-}$. Note that, the latter three parameters/columns are not relevant for the test but demonstrate the variety of the 3 raw NIRS signals.

Table 4.2 shows the smallest magnitudes ε for which the optical neuronal signal is significant, i.e. $\varepsilon > 1.96\sigma_s$ where $\sigma_s \triangleq$ empirical standard deviation of all samples (except for the modified one) inside the average segment. According to literature, the magnitude of optical neuronal signals is expected in the range of 0.01% – 0.24% relative to the magnitude of the raw NIRS signals (see Section 1.3). Conclusively, $\frac{100 \cdot 0.9}{1479.2} = 0.06\%$, $\frac{100 \cdot 0.5}{16687.23} = 0.003\%$ and $\frac{100 \cdot 0.5}{21298.89} = 0.0023\%$ proposes that in the study above, an optical neuronal signal should have been detected, if it had been existent in the raw NIRS signals.

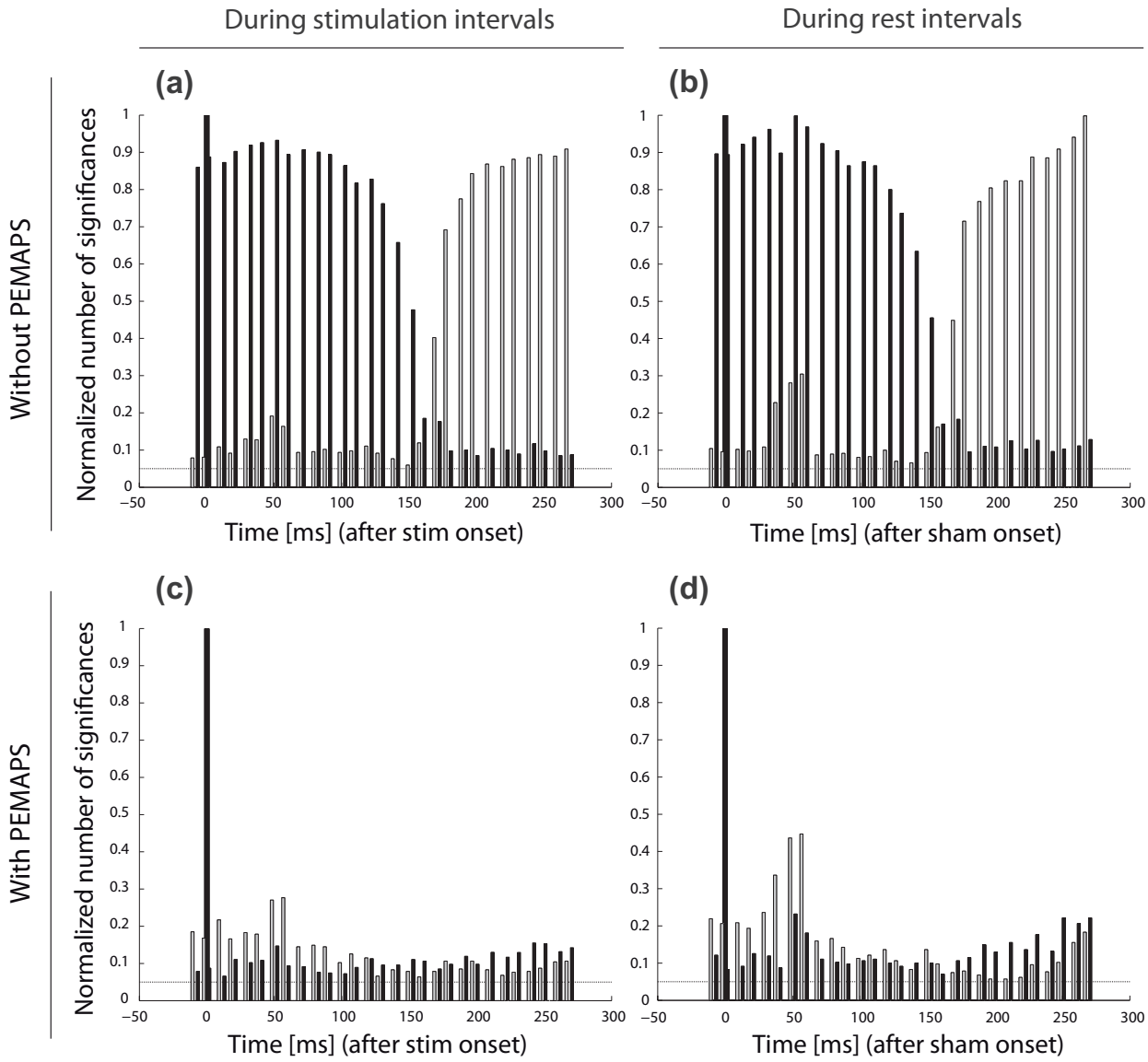

Figure 4.13: Distribution of significant ($p<0.05$) samples in the average segments over all data channels and measurements. The number of occurrences is normalised to the total number of data points which contribute to the histograms. The black bar at time 0 ms marks the beginning of a stimulation event. The other black bars are related to significant negative samples; gray bars are related to significant positive samples. The number of significances in plots (a) and (b) is higher than the number of significances in plots (c) and (d) showing the effectiveness of PEMAPS in reducing false positive data, importance of proper heartbeat removal and the necessity of comparing stimulation and rest intervals.

Chapter 5
An approach to detecting movement artefacts

The following algorithm may be used to adaptively detect artefacts in Fig. 5.1.

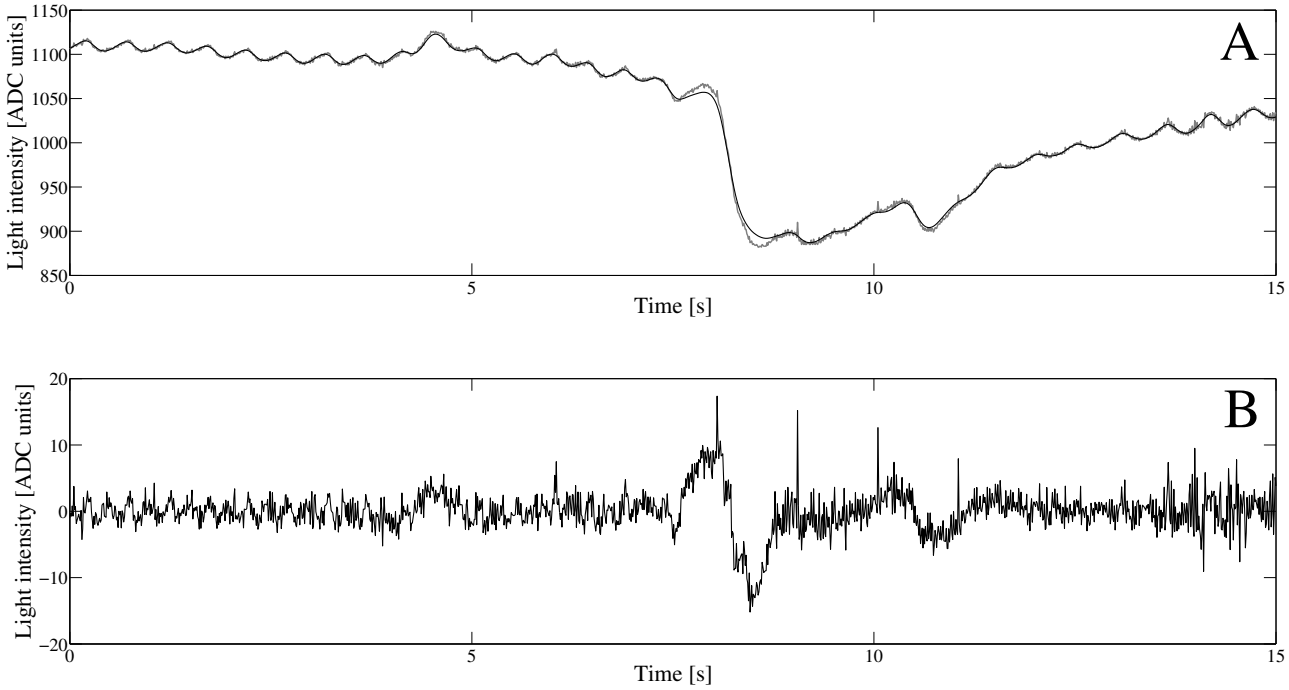

Figure 5.1: Example of a raw NIRS signal $\boldsymbol{y}$ (grey curve in A) with sampling rate 100 Hz with movement artefacts. The corresponding reconstructed heartbeat oscillation $\hat{\boldsymbol{x}}$ including the slow trend $\hat{\boldsymbol{A}}_{0,-}$, both computed with POETGaussian, is the black curve in A; the residual signal $\boldsymbol{y} - \hat{\boldsymbol{x}}$ is depicted in B. The signal values are given in "ADC units" since the NIRS instrumentation uses an ADC to digitise light intensity.

1. Compute the residual $\boldsymbol{z} = \boldsymbol{y} - \hat{\boldsymbol{x}}$ in plot B with POETGaussian or POETDiscretePhase.
2. Compute the empirical standard deviation $\sigma_{\boldsymbol{z}}$ of $\boldsymbol{z}$.
3. Define a threshold $1.5\sigma_{\boldsymbol{z}} \lessapprox \tau \lessapprox 2.5\sigma_{\boldsymbol{z}}$.
4. Every sample z_n in $\boldsymbol{z}$ higher than τ including some number of neighbouring samples are declared to belong to an artefact. This would probably detect the distorted region between 7 s and 9 s.
5. Create two segments of $\boldsymbol{z}$: $\boldsymbol{z}_1$ corresponds to the part of $\boldsymbol{z}$ between 0 s and 7 s, $\boldsymbol{z}_2$ corresponds to the part of $\boldsymbol{z}$ between 9 s and 15 s.
6. Execute steps 2–5 for both segments.

If this algorithm is too sensitive and declares too many regions to belong to an artefact, τ should be increased. Decreasing τ would help if the algorithm does not detect robustly enough.

Chapter 6
A framework for comparing and evaluating algorithms for signal separation

A framework for comparing and evaluating algorithms for signal separation has been developed and implemented in the scope of this thesis in [58]. The framework implements an interface for embedding signal separating, generating and evaluating algorithms.

To synthesise an oscillatory component, a real NIRS signal is band-pass filtered to obtain an estimate of the oscillatory component, then the periods of the latter are detected, and finally the relative changes from one sample to the next, defined as "features", are extracted based on the detected periods. To keep the size of the hidden markov model (HMM) low, the range of values of the features is quantised. An initial HMM is then trained based on the extracted quantised features. One oscillatory component per HMM is derived based on which a new oscillatory component is synthesised. Adding up all synthesised components and computer-generated noise results in a synthesised NIRS signal. An algorithm (called "Filter" in the framework) may now estimate the oscillatory components in the synthesised NIRS signal. The estimates may then be compared with the corresponding synthesised oscillatory components by using an "Evaluator". The latter comprises different cost functions.

The C++ project of this framework is called "SimNirs". To include a new plugin, i.e. a signal generator, a filter, or an evaluator,

1. place the new `.cpp` and `.hpp` files in the corresponding directory `./Sim/Generators`, `./Sim/Filters` or `./Sim/Eval` (paths are relative to the root directory of SimNirs),
2. adapt `subdir.mk` in the same directory where the new files were placed,
3. include the new `.hpp` file in `./Sim/Simcore.cpp` and add the corresponding `AddPlugin(...)` line in the method `SimCore::allocatePlugins()`.

To compile and run SimNirs under Ubuntu Linux,

1. install the packages "libqt4-dev" and "libqwt5-qt4",
2. ensure that the file `libqwt.so` in `/usr/lib/` exists or create a link with that name pointing to `/usr/lib/libqwt-qt4.so.5`,
3. execute `make` inside `./Sim`, and `qmake` followed by a `make` inside `./SimGUI`,
4. execute first `./Sim/Release/Sim` in one console and then `./SimGUI/SimGUI` in another console.

The specific package names vary between Linux distributions and releases.

EMD, POETDiscretePhase and TraditionalFilter (bandpass) have been included in SimNirs. Including also SSA might be a suggestive step.

In conclusion, SimNirs is ready to use, but no results have been computed yet. At the moment, the signal generator uses HMMs trained with bandpass filtered NIRS signals. Training the HMMs using unfiltered, low-noise raw NIRS signals would be more convenient.

Conclusions and outlook

In this thesis, an intuitive model for almost periodic signals and two algorithms for estimating the model parameters from noisy signals measured with continuous wave NIRS have been developed and implemented. The model explicitly incorporates the properties of almost periodic signals, and thus they can precisely be separated from the measured NIRS signal. A framework has been developed and implemented to (i) quantitatively assess the precision and compare different methods for signal separation based on synthesised NIRS signals, and (ii) calibrate each method individually with respect to the almost periodic signal to be separated. Synthesising authentic NIRS signals is a nontrivial key task which should be independent of the methods for signal separation. It might be convenient to (i) synthesise NIRS signals using physical models of arteries, hemodynamics and the cardiovascular system, or/and (ii) use real, low-noise NIRS signals for the synthesis.

Simultaneous NIRS and ECG measurements show that the proposed algorithms correctly estimate the heartbeat component. The algorithms have not been tested yet with the breathing component, since the latter is only sporadically observed in raw NIRS signals. Simultaneous NIRS and air flow measurements could help to reveal the circumstances under which measuring the breathing component with NIRS, if at all, is reproducible. First tests with low frequency oscillations (LFO), also Mayer waves, look promising but must be confirmed by further studies or evaluations based on the proposed framework. Since LFOs are defined through blood pressure, simultaneous NIRS and blood pressure measurements may help.

The proposed algorithms are limited to almost periodic signals with a strong fundamental frequency. The extension to signals with arbitrary energy distributions is currently being developed at the Signal and Information Processing Laboratory (ISI), ETH Zurich and is described in [59]. A problem is to identify the fundamental frequency when it and/or some of its harmonics are absent.

Based on the estimated model parameters, physiological questions may be investigated and answered, e.g. does the phase/amplitude of the heartbeat component/LFO synchronise to the stimulus events in functional experiments? Are the parameter estimates of the heartbeat component and the LFO related to light wavelength, source/detector distance/arrangement?

The proposed algorithms are fast and work with short signals. Thus they are well suited for real-time applications where low latencies are important, e.g. online monitoring of patients. In such applications, the forward messages derived from the past measured samples may be used as prior messages in the factor graphs arising from just acquired samples.

Although we have not detected optical neuronal signals in our functional study, the more efficient one of the algorithms significantly reduced the disturbing effect of the heartbeat component. In addition, simulation results propose that the optical neuronal signals should be detectable from estimates computed by the more efficient one of the algorithms.

Acronyms and symbols

ADC analog-to-digital converter
CSV comma-separated values
ECG electrocardiography
EEG electroencephalography
EMD empirical mode decomposition
EOF empirical orthogonal functions
HMM hidden markov model
HRV heart rate variability
ICA independent component analysis
IO input/output
IMF intrinsic mode function
LFO low frequency oscillation
NIRI near-infrared imaging
NIRS near-infrared spectroscopy
OT optical topography
OS operating system
PCA principal component analysis
PDF probability density function
PEMAPS parameter estimation of a model for almost periodic signals
POET physiological oscillation estimation tool
NaN not-a-number
SDL Simple DirectMedia Layer
SNR signal-to-noise ratio
SSA singular spectrum analysis
STL Standard Template Library
UML Unified Modelling Language

Bibliography

[1] M. Wolf, M. Ferrari, and V. Quaresima, "Progress of near-infrared spectroscopy and topography for brain and muscle clinical applications," *Journal of Biomedical Optics*, vol. 12, p. 062104, 2007.

[2] D. V. Haensse, *Changes in Cerebral Oxygenation in Response to Various Stimuli in Newborns as Measured by Functional Near-Infrared Spectroscopy*. PhD thesis, ETH Zurich, 2006.

[3] D. T. Delpy, M. Cope, P. van der Zee, S. Arridge, S. Wray, and J. Wyatt, "Estimation of optical pathlength through tissue from direct time of flight measurement," *Phys. Med. Biol.*, vol. 33, pp. 1433–1442, 1988.

[4] D. Haensse, P. Szabo, D. Brown, J.-C. Fauchère, P. Niederer, H.-U. Bucher, and M. Wolf, "A new multichannel near infrared spectrophotometry system for functional studies of the brain in adults and neonates," *Optics Express*, vol. 13, pp. 4525–4538, 2005.

[5] C. Julien, "The enigma of mayer waves: Facts and models," *Cardiovascular Research*, vol. 70, pp. 12–21, 2006.

[6] R. Stepnoski, A. LaPorta, F. Raccuia-Behling, G. E. Blonder, R. E. Slusher, and D. Kleinfeld, "Noninvasive detection of changes in membrane potential in cultured neurons by light scattering," *Proc. Natl. Acad. Sci. USA*, vol. 88, pp. 9382–9386, 1991.

[7] J. Lee and S. Kim, "Spectrum measurement of fast optical signal of neural activity in brain tissue and its theoretical origin," *NeuroImage*, vol. 51, pp. 713–722, 2010.

[8] G. Gratton, C. R. Brumback, B. A. Gordon, M. A. Pearson, K. A. Low, and M. Fabiani, "Effects of measurement method, wavelength, and source-detector distance on the fast optical signal," *NeuroImage*, vol. 32, pp. 1576–1590, 2006.

[9] M. A. Franceschini and D. A. Boas, "Noninvasive measurement of neuronal activity with near-infrared optical imaging," *NeuroImage*, vol. 21, pp. 372–386, 2004.

[10] M. Wolf, U. Wolf, J. H. Choi, R. Gupta, L. P. Safonova, L. A. Paunescu, A. Michalos, and E. Gratton, "Functional frequencydomain near-infrared spectroscopy detects fast neuronal signal in the motor cortex," *NeuroImage*, vol. 17, pp. 1868–1875, 2002.

[11] M. Wolf, U. Wolf, J. H. Choi, V. Toronov, L. A. Paunescu, A. Michalos, and E. Gratton, "Fast cerebral functional signal in the 100-ms range detected in the visual cortex by frequency-domain near-infrared spectrophotometry," *Psychophysiology*, vol. 40, pp. 521–528, 2003.

[12] J. Steinbrink, M. Kohl, H. Obrig, G. Curio, F. Syre, F. Thomas, H. Wabnitz, H. Rinneberg, and A. Villringer, "Somatosensory evoked fast optical intensity changes detected non-invasively in the adult human head," *Neurosci. Lett.*, vol. 291, pp. 105–108, 2000.

[13] J. Steinbrink, F. C. Kempf, A. Villringer, and H. Obrig, "The fast optical signal–robust or elusive when non-invasively measured in the human

adult?," *NeuroImage*, vol. 26, pp. 996–1008, 2005.
[14] A. Hyvärinen and E. Oja, "Independent component analysis: algorithms and applications," *Neural Networks*, vol. 13, no. 4–5, pp. 411–430, 2000.
[15] N. E. Huang, Z. Shen, S. R. Long, M. C. Wu, H. H. Shih, Q. Zheng, N. Yen, C. C. Tung, and H. H. Liu, "The empirical mode decomposition and the hilbert spectrum for nonlinear and non-stationary time series analysis," *Proc. R. Soc. Lond. A*, vol. 454, pp. 903–995, 1996.
[16] F. Scholkmann, "Signalverarbeitung in Bezug auf Untersuchung des Gehirns mit Licht: Entwicklung und Optimierung von Filtern und Charakterisierung von verschiedenen Signalkomponenten," diploma thesis, University Hospital Zurich / FH Isny, 2007.
[17] K. Pearson, "On lines and planes of closest fit to systems of points in space," *Philosophical Magazine*, vol. 2, pp. 559–572, 1901.
[18] S. Coyle, T. Ward, and C. Markham, "Physiological noise in near-infrared spectroscopy: Implications for optical brain computer interfacing," *Proc. 26th Annual International Conf. of the IEEE EMBS*, vol. 2, pp. 4540–4543, 2004.
[19] I. Tachtsidis, C. E. Elwell, T. S. Leung, C.-W. Lee, M. Smith, and D. T. Delpy, "Investigation of cerebral haemodynamics by near infrared spectroscopy in young healthy volunteers reveals posture dependent spontaneous oscillations," *Physiological Measurement*, vol. 25, pp. 437–445, 2004.
[20] H. Hassani, "Singular spectrum analysis: Methodology and comparison," *Journal of Data Science*, vol. 5, pp. 239–257, 2007.
[21] H.-A. Loeliger, J. Dauwels, J. Hu, S. Korl, L. Ping, and F. R. Kschischang, "The factor graph approach to model-based signal processing," *Proc. of the IEEE*, vol. 95, pp. 1295–1322, 2007.
[22] H.-A. Loeliger, "An introduction to factor graphs," *IEEE Signal Proc. Mag.*, vol. 21, pp. 28–41, 2004.
[23] S. Wray, M. Cope, D. T. Delpy, J. S. Wyatt, and E. O. Reynolds, "Characterization of the near-infrared absorption spectra of cytochrome aa3 and haemoglobin for the non-invasive monitoring of cerebral oxygenation," *Biochim. Biophys. Acta*, vol. 933, pp. 184–192, 1988.
[24] H. Obrig, M. Neufang, R. Wenzel, M. Kohl, J. Steinbrink, K. Einhäupl, and A. Villringer, "Spontaneous low frequency oscillations of cerebral hemodynamics and metabolism in human adults," *NeuroImage*, vol. 12, no. 6, pp. 623–639, 2000.
[25] I. Trajkovic and C. Wacha, "Ad hoc communication with handhelds," semestral thesis, ETH Zurich, 2006.
[26] T. F. of The European Society of Cardiology, T. N. A. S. of Pacing, and Electrophysiology, "Heart rate variability, standards of measurement, physiological interpretation, and clinical use," *European Heart Journal*, vol. 17, pp. 354–381, 1996.
[27] R. E. Kleiger, J. P. Miller, J. T. Bigger, and A. J. Moss, "Decreased heart rate variability and its association with increased mortality after acute myocardial infarction," *The American journal of cardiology*, vol. 59,

pp. 256–262, 1987.
[28] M. Malik, T. Farrell, T. Cripps, and A. J. Camm, "Heart rate variability in relation to prognosis after myocardial infarction: selection of optimal processing techniques," *European Heart Journal*, vol. 10, pp. 1060–1074, 1989.
[29] J. T. Bigger, J. L. Fleiss, R. C. Steinman, L. M. Rolnitzky, R. E. Kleiger, and J. N. Rottman, "Frequency domain measures of heart period variability and mortality after myocardial infarction," *Circulation*, vol. 85, pp. 164–171, 1992.
[30] I. Trajkovic, C. Reller, M. Wolf, and H.-A. Loeliger, "Modelling and filtering almost periodic signals by time-varying fourier series with application to near infrared spectroscopy," *Proc. 17th European Signal Proc. Conf. (EUSIPCO)*, pp. 632–636, 2009.
[31] F. Scholkmann, S. Spichtig, T. Muehlemann, and M. Wolf, "How to detect and reduce movement artifacts in near-infrared imaging using moving standard deviation and spline interpolation," *Physiological Measurement*, vol. 31, no. 5, pp. 649–662, 2010.
[32] J. M. Bland and D. G. Altman, "Statistical methods for assessing agreement between two methods of clinical measurement," *Lancet*, vol. 327, no. 8476, pp. 307–310, 1986.
[33] M. Merri, D. Farden, J. Mottley, and E. Titlebaum, "Sampling frequency of the electrocardiogram for spectral analysis of the heart rate variability," *IEEE Transactions on Biomedical Engineering*, vol. 37, no. 1, pp. 99–106, 1990.
[34] P. Stein, M. Bosner, R. Kleiger, and B. Conger, "Heart rate variability: a measure of cardiac autonomic tone," *American Heart Journal*, vol. 127, no. 5, pp. 1376–1381, 1994.
[35] R. Rauh, R. Limley, R. Bauer, M. Radespiel-Troger, and M. Mueck-Weymann, "Comparison of heart rate variability and pulse rate variability detected with photoplethysmography," *Proc. SPIE*, vol. 5474, pp. 115–126, 2004.
[36] S. Lu, H. Zhao, K. Ju, K. Shin, M. Lee, K. Shelley, and K. H. Chon, "Can photoplethysmography variability serve as an alternative approach to obtain heart rate variability information?," *Journal of Clinical Monitoring and Computing*, vol. 22, no. 1, pp. 23–29, 2008.
[37] G. Lu, F. Yang, J. A. Taylor, and J. F. Stein, "A comparison of photoplethysmography and ecg recording to analyse heart rate variability in healthy subjects," *Journal of Medical Engineering & Technology*, vol. 33, no. 8, pp. 634–641, 2009.
[38] N. Selvaraj, A. Jaryal, J. Santhosh, K. K. Deepak, and S. Anand, "Assessment of heart rate variability derived from finger-tip photoplethysmography as compared to electrocardiography," *Journal of Medical Engineering & Technology*, vol. 32, no. 6, pp. 479–484, 2008.
[39] T. Katura, N. Tanaka, A. Obata, H. Sato, and A. Maki, "Quantitative evaluation of interrelations between spontaneous low-frequency oscillations

in cerebral hemodynamics and system- ic cardiovascular dynamics," *NeuroImage*, vol. 31, pp. 1592–1600, 2006.

[40] I. Trajkovic, *Modelling and filtering almost periodic signals by time-varying fourier series with application to near-infrared spectroscopy.* PhD thesis, ETH Zurich, 2010.

[41] A. V. Medvedev, J. Kainerstorfer, S. V. Borisov, R. L. Barbour, and J. VanMeter, "Event-related fast optical signal in a rapid object recognition task: Improving detection by the independent component analysis," *Brain Research*, vol. 1236, pp. 145–158, 2008.

[42] C. Y. Tse, B. A. Gordon, M. Fabiani, and G. Gratton, "Frequency analysis of the visual steady-state response measured with the fast optical signal in younger and older adults," *Biological Psychology*, vol. 85, no. 1, pp. 79–89, 2010.

[43] M. Wolf, G. Morren, D. Haensse, T. Karen, U. Wolf, J. C. Fauchère, and H. U. Bucher, "Near infrared spectroscopy to study the brain: an overview," *Opto-Electronics Review*, vol. 16, no. 4, pp. 413–419, 2008.

[44] R. B. Saager and A. J. Berger, "Direct characterization and removal of interfering absorption trends in two-layer turbid media," *Journal of the Optical Society of America A*, vol. 22, no. 9, pp. 1874–1882, 2005.

[45] H. Herbert and H. Jaspers, "Report of the committee on methods of clinical examination in electroencephalography: 1957," *Electroencephalography and Clinical Neurophysiology*, vol. 10, no. 2, pp. 370–375, 1958.

[46] L. Kocsis, P. Herman, and A. Eke, "The modified beer–lambert law revisited," *Physics in medicine and biology*, vol. 51, p. N91, 2006.

[47] H. Zhao, Y. Tanikawa, F. Gao, Y. Onodera, A. Sassaroli, K. Tanaka, and Y. Yamada, "Maps of optical differential pathlength factor of human adult forehead, somatosensory motor and occipital regions at multi-wavelengths in nir," *Physics in medicine and biology*, vol. 47, p. 2075, 2002.

[48] G. Gratton and P. Corballis, "Removing the heart from the brain: Compensation for the pulse artifact in the photon migration signal," *Psychophysiology*, vol. 32, no. 3, pp. 292–299, 1995.

[49] I. Trajkovic, C. Reller, and M. Wolf, "Modelling and filtering of physiological oscillations in near-infrared spectroscopy by time-varying fourier series," *Proc. International Society on O2 Transport to Tissue (ISOTT)*, 2010. to be published.

[50] J. F. Echallier, F. Perrin, and J. Pernier, "Computer-assisted placement of electrodes on the human head," *Electroencephalography and Clinical Neurophysiology*, vol. 82, no. 2, pp. 160–163, 1992.

[51] M. M. Plichta, M. J. Herrmann, C. G. Baehne, A. C. Ehlis, M. M. Richter, P. Pauli, and A. Fallgatter, "Event-related functional near-infrared spectroscopy (fnirs): Are the measurements reliable?," *NeuroImage*, vol. 31, no. 1, pp. 116–124, 2006.

[52] W. C. M. Machielsen, S. A. R. B. Rombouts, F. Barkhof, P. Scheltens, and M. P. Witter, "Fmri of visual encoding: reproducibility of activation," *Human Brain Mapping*, vol. 9, no. 3, pp. 156–164, 2000.

[53] I. and T. Benavente, Pilar, T. Natividad, Y. Valentín, and M. Oliván, "Flash visually evoked potentials in the newborn and their maturation during the first six months of life," *Documenta Ophthalmologica*, vol. 110, no. 2, pp. 255–263, 2005.

[54] J. Sarnthein, M. Andersson, M. B. Zimmermann, and D. Zumsteg, "High test-retest reliability of checkerboard reversal visual evoked potentials (vep) over 8 months," *Clinical Neurophysiology*, vol. 120, no. 10, pp. 1835–1840, 2009.

[55] H. Ikeda, H. Nishijo, K. Miyamoto, R. Tamura, S. Endo, and T. Ono, "Generators of visual evoked potentials investigated by dipole tracing in the human occipital cortex," *Neuroscience*, vol. 84, pp. 723–739, 1998.

[56] B. Török, M. Meyer, and H. Wildberger, "The influence of pattern size on amplitude, latency and wave form of retinal and cortical potentials elicited by checkerboard pattern reversal and stimulus onset-offset," *Electroencephalography and Clinical Neurophysiology/Evoked Potentials Section*, vol. 84, pp. 13–19, 1992.

[57] J. V. Odom, M. Bach, C. Barber, M. Brigell, M. F. Marmor, A. P. Tormene, G. E. Holder, and Vaegan, "Visual evoked potentials standard (2004)," *Documenta Ophthalmologica*, vol. 108, pp. 115–123, 2004.

[58] S. Simonet, "Signalverarbeitung in Bezug auf Untersuchung des Gehirns mit Licht: Design und Implementierung eines Frameworks zur Synthese von NIRS Signalen," semestral thesis, ETH Zurich, 2009.

[59] C. Reller, I. Trajkovic, and H.-A. Loeliger, "A short-time fourier series for approximately periodic signals," in preparation.

Publications

I. Trajkovic, C. Reller, M. Wolf, and H.-A. Loeliger, "Modelling and Filtering Almost Periodic Signals by Time-Varying Fourier Series with Application to Near-Infrared Spectroscopy," *Proceedings of the 17th European Signal Processing Conference (EUSIPCO)*, pp. 632–636, 2009.

I. Trajkovic, C. Reller, and M. Wolf, "Modelling and Filtering of Physiological Oscillations in Near-Infrared Spectroscopy by Time-Varying Fourier Series," submitted to *Advances in Experimental Medicine and Biology*, accepted for publication.

I. Trajkovic, F. Scholkmann, and M. Wolf, "Estimating and Validating the Interbeat Intervals of the Heart Using Near-Infrared Spectroscopy on the Human Forehead," submitted to *Journal of Biomedical Optics (JBO)*, accepted for publication.

M. Biallas, I. Trajkovic, F. Scholkmann, C. Hagmann, and M. Wolf, "How to conduct studies with neonates combining near-infrared imaging and electroencephalography," submitted to *Advances in Experimental Medicine and Biology*, accepted for publication.

M. Biallas, I. Trajkovic, C. Hagmann, F. Scholkmann, L. Holper, A. Beck, C. Jenny, M. Wolf, "Multimodal recording of brain activity in term newborn infants during photic stimulation by near-infrared spectroscopy and electroencephalography" to be submitted to *NeuroImage*.

M. Biallas, I. Trajkovic, D. Haensse, V. Marcar and M. Wolf, "Reproducibility and sensitivity of detecting brain activity by simultaneous electroencephalography and near-infrared imaging," to be submitted to *Experimental Brain Research*.

C. Reller, I. Trajkovic, and H.-A. Loeliger, "A Short-Time Fourier Series for Approximately Periodic Signals," in preparation.

F. Scholkmann, S. Spichtig, I. Trajkovic, M. Biallas, M. Wolf, "Using ensemble empirical mode decomposition (EEMD) to extract evoked hemodynamic responses from hemodynamic signals measured with functional near-infrared spectroscopy," in preparation for *Biomedical Signal Processing and Control*.

C. Jenny, M. Biallas, I. Trajkovic, J.-C. Fauchère, H.-U. Bucher, M. Wolf, "Reproducibility of cerebral tissue oxygen saturation measurements by near infrared spectroscopy in newborn infants," submitted to *JBO*, accepted for publication.

F. Scholkmann, O. D. Sieber, L. Holper, I. Trajkovic, M. Wolf, "The fractality of spontaneous cerebral hemodynamic oscillations is body posture dependent," in preparation for *Communications in Nonlinear Science and Numerical Simulation*.

Danksagungen

Diese Menschen hatten Einfluss auf mich und meine Doktorarbeit auf verschiedenste Weise. Dies soll nicht in Vergessenheit geraten.

Prof. Markus Rudin hat mir an den Progress Reports stets Schlüsselfragen gestellt und wertvolle Anmerkungen gegeben. Ich danke ihm für diese gute Betreuung.

PD Martin Wolf unterstützte mich stets bei meinen wissenschaftlichen Ambitionen. Seine optimistische Einstellung und nette Art hat vieles erleichtert. Vielen Dank Martin.

Prof. Hans-Andrea Loeliger hat mir am Anfang den richtigen Anstoss gegeben um eine Brücke zu schlagen zwischen modellgestützter Signalverarbeitung und der Nahinfrarotspektroskopie. Seine Begeisterung für das Fach war ansteckend. Ich danke ihm für die konstruktiven Sitzungen voller Ideen und Elan.

Christoph Reller kritisiert so konstruktiv wie sonst niemand. Er steckt voller Ideen und ist sehr hilfsbereit. Die wirklich coole Zeit in Glasgow bleibt mir in Erinnerung. Vielen Dank für alles Christoph und viel Erfolg bei deiner Doktorarbeit.

Prof. Hans-Ulrich Bucher danke ich für die Bereitstellung der Infrastruktur und die Unterstützung bei der Neugeborenenstudie.

Thomas Mühlemann danke ich für die Bierabende und die Joggingrunden. Manche waren still, manche jedoch voller Diskussionen über verschiedenste Themen.

Felix Scholkmann, alias chairman, danke ich für den grossen Einsatz während seiner Diplomarbeit. Das Musizieren an der ISOTT 2010 mit Felix war grossartig. Ich wünsche ihm viel Erfolg bei seiner Doktorarbeit.

Martin Biallas und ich haben unzählige gemeinsame Stunden beim Messen und Paper schreiben verbracht. Er ist sehr realitätsbewusst und gleichzeitig kommuniziert er auf eine sehr angenehme Weise. Danke Martin und viel Erfolg bei deiner Doktorarbeit.

Juan Mata Pavia alias Huanče vermochte mich mit seiner sympatischen Art immer wieder zum lachen bringen. Ich wünsche ihm viel Erfolg bei seiner Doktorarbeit.

Daniel Haensse danke ich, dass er den fachlichen Austausch mit den

Leuten vom ISI eingeleitet hat. Die Badeveranstaltungen in der Limmat bleiben unvergessen.

Andreas Metz sorgte immer für lockere Unterhaltung an diversen Bierabenden and Anlässen. Ich wünsche ihm viel Erfolg bei seiner Doktorarbeit.

meine Freundin Lisa danke ich für die Geduld und das Verständnis vor allem gegen Ende meiner Arbeit. Ihr Umgang vermochte mich auch in stressigen Zeiten entspannen.

meine Eltern gilt ein besonderer Dank, weil sie mich bei (fast) all meinen Vorhaben bis jetzt unterstützt haben. Die leckeren Speisen meiner Mutter gehören in ein Meister-Kochbuch.

Cornelia Hagmann und *Andreas Beck* haben Martin Biallas und mir sehr geholfen bei der Rekrutierung von Eltern für unsere Studie mit Neugeborenen. Ich danke euch dafür herzlich.

Bryn Lloyd sei gedankt für das Durchlesen meines ersten Papers. Die Lauftrainings waren stets unterhaltsam.

Sonja Spichtig danke ich für die spannenden Diskussionen über die Wissenschaft allgemein während den seelengleichgewichterhaltenden Joggingrunden.

Clemens Wacha sei gedankt für die nützlichen C++ bibliotheken.

Lukas Bolliger und *Raphael Zimmermann* danke ich für die unterhaltsamen Joggingrunden.

Damien de Courten danke ich für die harten aber auch lustigen Lauf/Klettertrainings am Ütliberg. Ich wünsche ihm viel Erfolg bei seiner Doktorarbeit und den Wettkämpfen in eisiger Kälte.

Frank Assenmacher danke ich für seine Abschlussarbeit, die zwar keine Verwendung fand innerhalb dieser Arbeit, jedoch sehr gut war.

Brigitte Koller danke ich für ihre Spontanität. Mit ihr gingen die Gesprächsthemen am Mittagstisch nie aus.

Dorli und Iris danke ich für ihre sofortige Hilfe bei bürotechnischen und administratorischen Angelegenheiten. Danke auch für den einen oder anderen lustigen Spruch.

Curriculum Vitae

I was born on March 3, 1980 in Belgrade, Socialist Federal Republic of Yugoslavia. I lived and completed two years of elementary school in Belgrade.

In 1989, I moved with my family to Ennetbaden AG, Switzerland.

In summer 2000, I graduated with the Matura Typus C from the Kantonsschule Baden. In the same year, I began my studies at the ETH Zurich.

I received the diploma in electrical engineering and information technology from the ETH Zurich in May, 2007.

Two months later, I started my dissertation in the Biomedical Optics Research Laboratory (BORL), headed by PD Dr. Martin Wolf, under the supervision of Prof. Dr. Markus Rudin and PD Dr. Martin Wolf. My work was focused on signal separation in near-infrared spectroscopy based on factor graphs and message passing algorithms. Since the latter are one of the main research fields of Prof. Dr. Hans-Andrea Loeliger, this work was conducted in cooperation with Professor Loeliger and Christoph Reller who are both with the Signal and Information Processing Laboratory (ISI), ETH Zurich.